



2023 CCF国际AIOps挑战赛决赛  
暨“大模型时代的AIOps”研讨会

# SRE-Copilot: 基于LLM的多场景智能运维

战队：SRE-Copilot      王宁

字节跳动

主办单位：中国计算机学会（CCF）、清华大学、中国建设银行股份有限公司、南开大学

承办单位：中国计算机学会互联网专委会、清华大学计算机科学与技术系、中国建设银行股份有限公司运营数据中心、南开大学软件学院、北京必示科技有限公司

赞助单位：华为技术有限公司、国网宁夏电力有限公司电力科学研究院、软通动力信息技术（集团）股份有限公司

# 目录

# CONTENTS

第一章节 团队介绍

第二章节 选题说明

第三章节 方案阐述

第四章节 效果展示

## 第一章节

# 团队介绍

# 团队介绍： SRE-Copilot



沿着成本、稳定性、效率、服务  
四条主线，为SRE提供数据化与  
智能化支持，协同和赋能SRE从  
DataOPS向AIOps、ChatOPS转变，  
是我们一直努力的方向。

ByteDance

**字节跳动**  
**基础架构-SRE-数据化**  
**@SRE-Copilot**

## 第二章节

# 选题说明

- 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（系统架构复杂，数据量庞大，数据种类多等）

## 系统架构图

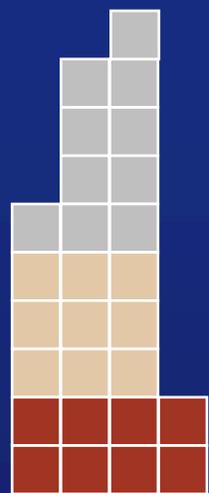


## 数据描述

- 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。

## 痛点

系统复杂，依赖繁多，  
海量数据

01

数据无标注  
训练成本高

02

接入/维护成本

03

新的故障推理

04

交互使用成本

05

## 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

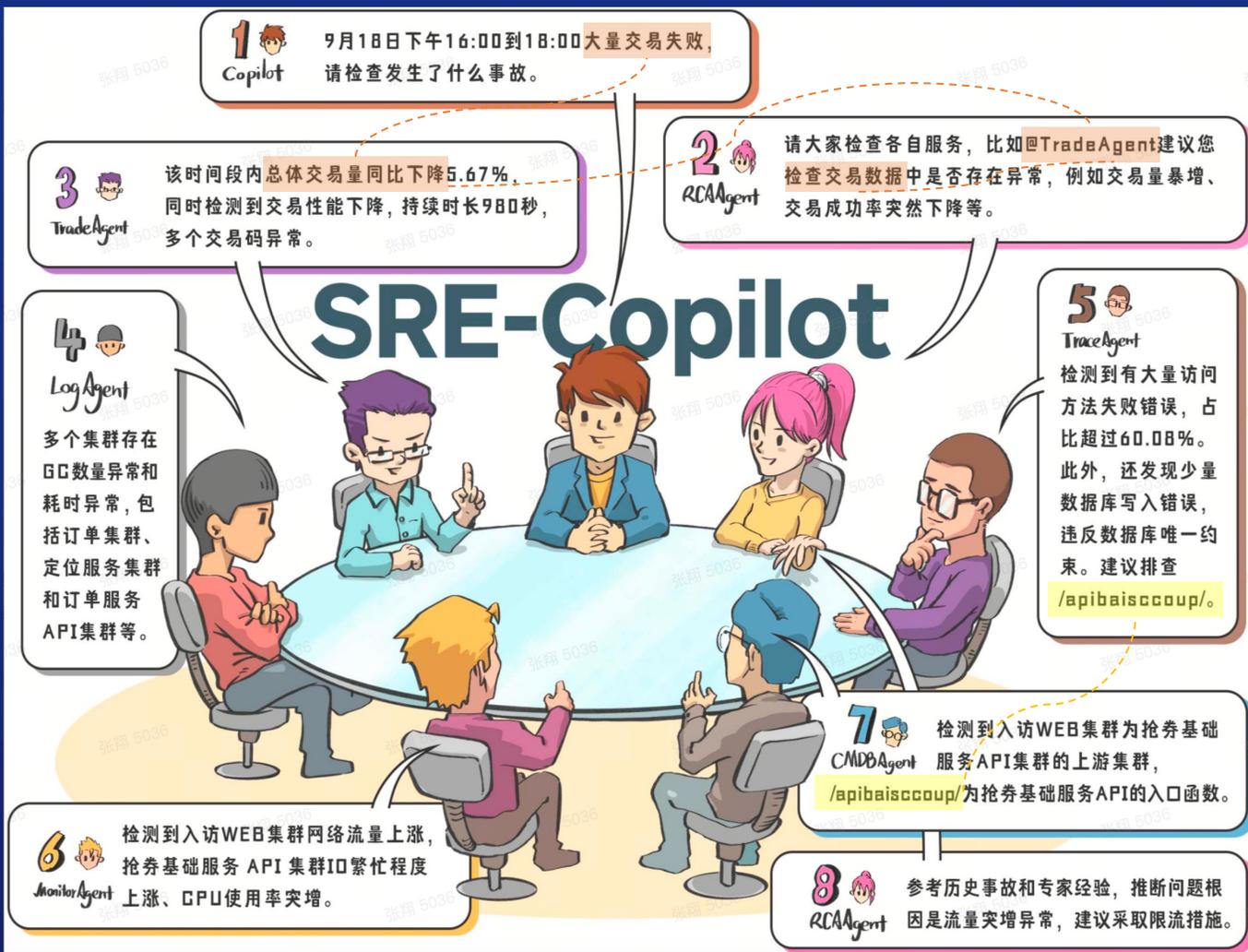
自然语言交互，更加智能，可以开放给更多用户

## 第三章节

# 方案阐述

# 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
- 降低噪音, 鲁棒性更高: 异常检测关注异常时候多个组件的表现形态, 而非根据单一组件判断是否异常

# 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 详见附录

# 辅助运维能力：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

## 故障分类

@WorkflowAgent SRE线上排查问题的通用流程是怎样的

WorkflowAgent: 我猜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:

1. 确认问题: 确定用户遇到的问题, 并记录下来。
2. 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。
3. 分析问题: 分析收集到的信息, 确定问题的根本原因。
4. 确定解决方案: 根据问题的分析结果, 确定可能的解决方案。
5. 实施解决方案: 选择最合适的解决方案, 并实施它。
6. 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。
7. 记录问题: 记录问题的解决情况, 并向用户提供解决问题的反馈。
8. 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。

根据目前的插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 = fsutil.df(disk)
free_space = disk_space[0].split()[1]

# 获取磁盘所有文件信息
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 = fsutil.df(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"清理后的磁盘空间: {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, 然后执行以下操作:
1. 使用 'fsutil.df' 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' 函数将文件删除。
3. 使用 'fsutil.df' 函数重新获取磁盘的容量, 并返回清理后的剩余空间。

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

1. 性能指标故障最早的开始时间: 2023-09-14 00:23:00
2. 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。
3. 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

1. 性能指标故障: nginx\_01、Weblogic\_07、Weblogic\_09
2. 调用链异常: 未提及具体集群
3. 日志异常: 未提及具体集群

### 三、各类告警中的异常具体表现

1. 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。
2. 调用链异常: 大量调用链中存在点丢失问题错误, 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。
3. 日志异常: 未提及具体异常表现。

### 四、此次故障的根因是什么, 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

1. 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。
2. 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, 也有可能与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。
3. 日志异常: 未提及具体根因, 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

1. 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。
2. 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。
3. 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。

@MonitorAgent, 请看一下2023-09-20 03:00:00到2023-09-20 04:00:00异常的机器性能指标

MonitorAgent: 我猜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

TraceAgent: 我猜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(basic/cr)类型的trace: /basic/cr/status

Mission Complete

## 运维可视化

@QAAgent, 你知道aiops挑战赛的主办方是谁吗

QAAgent: 我猜QAAgent能够解决您的问题 您要检测的开始时间为 2023-11-19 17:10:12, 截止时间为 2023-11-19 17:10:12

QAAgent: 很抱歉, 我没有找到关于AIOps挑战赛主办方的具体信息。

@QAAgent, 你知道aiops挑战赛的主办方是谁吗, 请参考local的data\_source

QAAgent: 我猜QAAgent能够解决您的问题 您要检测的开始时间为 2023-11-19 16:10:44, 截止时间为 2023-11-19 17:10:44

QAAgent: 是的, 我知道。本届CCF国际AIOps挑战赛由中国计算机学会 (CCF)、清华大学、中国建设银行股份有限公司联合主办; CCF互联网专委会、清华大学计算机科学与技术系、中国建设银行股份有限公司运营数据中心、北京必示科技有限公司承办。

workflows生成 (运维计划)

故障自愈+代码生成

故障报告

知识库问答



## 第四章

# 效果展示

# 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



**Oncall Agents**

- MonitorAgent
- CMDBAgent
- TraceAgent
- LogAgent
- TradeAgent
- Copilot
- RCAAgent

## SRE-COPLOT

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。

有什么问题吗?

## 参赛主要运维能力：

- ★ **异常检测**：支持多种数据类型，灵活拓展，通过多agent协同，整合不同平台数据，可以极大缩短MTTR
- ★ **根因定位**：无监督，对于未知故障可推理，可编码专家经验/历史故障，对已知故障准确率高

## 参赛辅助运维能力：

- **workflow生成**：对用户的需求进行分析，生成自然语言的工作流，并选择合适组件，构造可执行的工作流
- **代码生成**：基于用户的提示生成代码。例如，将复杂脚本的调试开发时间从几小时缩短到几分钟
- **故障报告**：生成完善的事报告，利用大语言模型显著提升故障诊断报告质量。方便团队积累经验和知识
- **故障分类**：根据故障描述对故障进行分类
- **运维可视化**：通过自然语言交互，自动执行简易的数据查询/分析，对故障数据进行可视化
- **知识库问答**：基于本地知识库进行私域知识问答，可以提供给不同行业/企业
- **故障自愈**：流程自动化，让运维人员集中精力，无需频繁切换上下文，确保响应和处理的及时性和准确性

## 展 望

更强大的通用大模型：chatglm3已经能达到如此效果，未来接入更强大的通用模型，效果会更加理想

更强大的专业大模型：针对metrics/log的专业大模型，会让异常检测插件更强大

丰富的功能插件市场：不断将运维场景智能化，欢迎业界一起完善生态



2023 CCF国际AIOps挑战赛决赛暨“大模型时代的AIOps”研讨会

# THANKS

# 附录

```
#####
#                               #
#           Welcome to LINUX     #
#                               #
#           All connections are  #
#           monitored and recorded. #
#           Disconnect IMMEDIATELY #
#           if you are not an #
#           authorized user!     #
#####
Web console: https://VM-2-133-kylin:9090/ or https://10.3.2.133:9090/

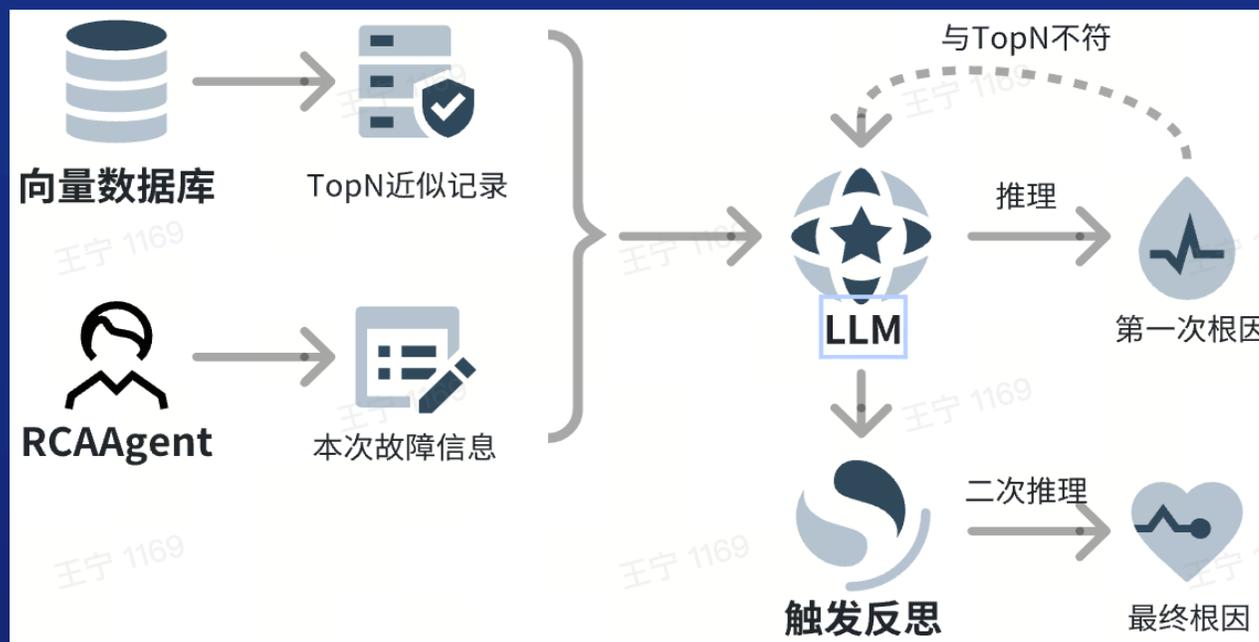
Last login: Sun Dec 10 15:02:54 2023 from 10.0.3.16
(base) root@VM-2-133-kylin:~>cd /user_data/aiops/aiops_challenge_2023/
(base) root@VM-2-133-kylin:/user_data/aiops/aiops_challenge_2023>sh start.sh
Hello, aiops_challenge_2023!
No sentence-transformers model found with name /user_data/embedding_model/GanymedeNil_text2vec-large-chinese. Creating a new one with MEAN pooling.
```

**SRE-COPILLOT**

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。

请输入您的问题：2023-09-17 09:18:00到2023-09-17 09:41:00，有什么问题

显示工具栏



# 推理新故障的例子

RCAAgent: @TraceAgent, 检测调用链相关信息, 定位时间范围内是否有问题。

TraceAgent: 总结后的结果为: 访问方法失败和调用链中存在点表失问题是当前系统面临的主要问题, 分别占比56.04%和43.85%。需要针对这些问题进行进一步的分析处理。

RCAAgent: @TradeAgent, 检测到交易大量失败, 需要分析交易相关指标, 如交易量、交易成功率、交易时长等。@MonitorAgent, 检测到集群维度的信息, 需要分析集群的机器是否存在故障, 以及nginx集群、ES集群、kafka集群、Redis集群、Oracle集群等是否存在问题。  
@LogAgent, 检测到日志是否有异常, 需要分析gc日志和access日志是否存在问题。

TradeAgent: 在该时间段内, 共有6次异常检测到交易数据性能下降, 具体如下:

- 交易数据性能下降异常, 开始时间为2023-09-19 16:45:40, 持续时长70秒, 影响的交易码有A0010, A0001, A0004, A0005, A0007。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:49:30, 持续时长30秒, 影响的交易码有A0004, A0010, A0005, A0007。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:51:00, 持续时长30秒, 影响的交易码有A0010, A0005, A0004, A0001。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:53:10, 持续时长30秒, 影响的交易码有A0010, A0001, A0005。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:50:20, 持续时长20秒, 影响的交易码有A0010, A0022, A0004。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:47:10, 持续时长10秒, 影响的交易码有A0004, A0001。

RCAAgent: @LogAgent, 该时间段内有6次异常, 具体如下:

- 交易数据性能下降异常, 开始时间为2023-09-19 16:45:40, 持续时长70秒, 影响的交易码有A0010, A0001, A0004, A0005, A0007。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:49:30, 持续时长30秒, 影响的交易码有A0004, A0010, A0005, A0007。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:51:00, 持续时长30秒, 影响的交易码有A0010, A0005, A0004, A0001。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:53:10, 持续时长30秒, 影响的交易码有A0010, A0001, A0005。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:50:20, 持续时长20秒, 影响的交易码有A0010, A0022, A0004。
- 交易数据性能下降异常, 开始时间为2023-09-19 16:47:10, 持续时长10秒, 影响的交易码有A0004, A0001。

根据异常信息, 可能需要对A0010交易码进行性能监控和排查, 以找出性能下降的原因。同时, 也需要对A0004交易码进行性能监控和排查, 以防止类似异常再次发生。

LogAgent: 当前时间, 各个集群的日志均未发现异常, 可以进一步排查其他数据来源。

RCAAgent: @MonitorAgent, 可以尝试查看各个集群的访问日志, 分析访问失败的原因。

MonitorAgent: 总结后的结果如下:

- 入访WEB集群存在TCP套接字异常和磁盘空间不足;
- 基础服务集群存在TCP套接字异常、网络流量上涨、内存使用率突增;
- 抢券公共库API集群存在TCP套接字异常、JAVA进程GC异常、内存使用率突增、CPU使用率突增;
- 批处理集群存在磁盘空间不足。

CMDBAgent: 定位到方法或者集群/order/值得进一步检查

RCAAgent: 检索到以下3条专家经验/历史case可能相符, 结果如下:

专家经验: Traffic, 表现: 入网集群nginx等流量相关指标突增,调用链存在丢失点, 措施LimitTraffic

专家经验: javaGC, 表现: JAVA进程GC异常, GC overhead limit exceeded错误, gc日志异常, 频繁GC, 措施OptimizeJVMSettings

专家经验: CacheOverflow, 表现: Redis集群缓存占用过高, 导致新的写入请求失败, 措施IncreaseCacheSize

RCAAgent: 参考历史事故和专家经验, 推断问题根因为PerformanceDrop, 建议采取Reboot措施

RCAAgent: @MonitorAgent, 请检查2023-08-20 07:00:00到08-20 08:00:00期间的集群状态, 特别是Weblogic集群、Nginx集群、ES集群、Kafka集群、Redis集群和Oracle集群。

MonitorAgent: 总结后的结果为: 所有检测项目均未发现异常信息。

RCAAgent: @TraceAgent, 请检测调用链相关信息, 定位是否有问题。

TraceAgent: 总结后的结果为: 在访问过程中出现故障, 占比38.49%, 同时伴有少量数据库写入错误, 违反数据库唯一约束。

RCAAgent: @TradeAgent, 检测交易相关指标出现问题时, 需要排查交易大量失败和交易耗时长的的问题。@LogAgent, 需要检测gc日志和access日志是否存在异常。

TradeAgent: 在该时间段内, 总体交易量同比下降, 相较于前一天下降了5.39%。

RCAAgent: @LogAgent, 需要检测该时间段内的access日志, 查看是否有异常交易量的情况。

LogAgent: 总结后的结果为: 订单集群的gc日志存在异常, 异常集群为Weblogic\_31, 主要表现为gc日志的GC数量异常。

CMDBAgent: 定位到方法或者集群/gc/, /, Weblogic\_31值得进一步检查

RCAAgent: 检索到以下3条专家经验/历史case可能相符, 结果如下:

专家经验: javaGC, 表现: JAVA进程GC异常, GC overhead limit exceeded错误, gc日志异常, 频繁GC, 措施OptimizeJVMSettings

专家经验: Traffic, 表现: 入网集群nginx等流量相关指标突增,调用链存在丢失点, 措施LimitTraffic

专家经验: CacheOverflow, 表现: Redis集群缓存占用过高, 导致新的写入请求失败, 措施IncreaseCacheSize

RCAAgent: 参考历史事故和专家经验, 推断问题根因为CacheOverflow, 通常的表现表现为Redis集群缓存占用过高, 导致新的写入请求失败, 建议采取IncreaseCacheSize措施

# 推理新故障的例子

RCAAgent: @TraceAgent, 检测调用链相关信息, 定位时间范围内是否有问题。

TraceAgent: 总结后的结果为: 访问方法失败和调用链中存在点表失问题是当前系统面临的主要问题, 分别占比56.04%和43.85%。需要针对这些问题进行进一步的分析处理。

RCAAgent: @TradeAgent, 检测到交易大量失败, 需要分析交易相关指标, 如交易量、交易成功率、交易时长等。@MonitorAgent, 检测到集群维度的信息, 需要分析集群的机器是否存在故障, 以及nginx集群、ES集群、kafka集群、Redis集群、Oracle集群等是否存在问题。  
@LogAgent, 检测到日志是否有异常, 需要分析gc日志和access日志是否存在问题。

TradeAgent: 在该时间段内, 共有6次异常检测到交易数据性能下降, 具体如下:

1. 交易数据性能下降异常, 开始时间为2023-09-19 16:45:40, 持续时长70秒, 影响的交易码有A0010, A0001, A0004, A0005, A0007。
2. 交易数据性能下降异常, 开始时间为2023-09-19 16:49:30, 持续时长30秒, 影响的交易码有A0004, A0010, A0005, A0007。
3. 交易数据性能下降异常, 开始时间为2023-09-19 16:51:00, 持续时长30秒, 影响的交易码有A0010, A0005, A0004, A0001。
4. 交易数据性能下降异常, 开始时间为2023-09-19 16:53:10, 持续时长30秒, 影响的交易码有A0010, A0001, A0005。
5. 交易数据性能下降异常, 开始时间为2023-09-19 16:50:20, 持续时长20秒, 影响的交易码有A0010, A0022, A0004。
6. 交易数据性能下降异常, 开始时间为2023-09-19 16:47:10, 持续时长10秒, 影响的交易码有A0004, A0001。

RCAAgent: @LogAgent, 该时间段内有6次异常, 具体如下:

1. 交易数据性能下降异常, 开始时间为2023-09-19 16:45:40, 持续时长70秒, 影响的交易码有A0010, A0001, A0004, A0005, A0007。
2. 交易数据性能下降异常, 开始时间为2023-09-19 16:49:30, 持续时长30秒, 影响的交易码有A0004, A0010, A0005, A0007。
3. 交易数据性能下降异常, 开始时间为2023-09-19 16:51:00, 持续时长30秒, 影响的交易码有A0010, A0005, A0004, A0001。
4. 交易数据性能下降异常, 开始时间为2023-09-19 16:53:10, 持续时长30秒, 影响的交易码有A0010, A0001, A0005。
5. 交易数据性能下降异常, 开始时间为2023-09-19 16:50:20, 持续时长20秒, 影响的交易码有A0010, A0022, A0004。
6. 交易数据性能下降异常, 开始时间为2023-09-19 16:47:10, 持续时长10秒, 影响的交易码有A0004, A0001。

根据异常信息, 可能需要对A0010交易码进行性能监控和排查, 以找出性能下降的原因。同时, 也需要对A0004交易码进行性能监控和排查, 以防止类似异常再次发生。

LogAgent: 当前时间, 各个集群的日志均未发现异常, 可以进一步排查其他数据来源。

RCAAgent: @MonitorAgent, 可以尝试查看各个集群的访问日志, 分析访问失败的原因。

MonitorAgent: 总结后的结果如下:

1. 入访WEB集群存在TCP套接字异常和磁盘空间不足;
2. 基础服务集群存在TCP套接字异常、网络流量上涨、内存使用率突增;
3. 抢券公共库API集群存在TCP套接字异常、JAVA进程GC异常、内存使用率突增、CPU使用率突增;
4. 批处理集群存在磁盘空间不足。

CMDBAgent: 定位到方法或者集群/order/值得进一步检查

RCAAgent: 检索到以下3条专家经验/历史case可能相符, 结果如下:

专家经验: Traffic, 表现: 入网集群nginx等流量相关指标突增,调用链存在丢失点, 措施LimitTraffic

专家经验: javaGC, 表现: JAVA进程GC异常, GC overhead limit exceeded错误, gc日志异常, 频繁GC, 措施OptimizeJVMSettings

专家经验: CacheOverflow, 表现: Redis集群缓存占用过高, 导致新的写入请求失败, 措施IncreaseCacheSize

RCAAgent: 参考历史事故和专家经验, 推断问题根因为PerformanceDrop, 建议采取Reboot措施

RCAAgent: @MonitorAgent, 请检查2023-08-20 07:00:00到08-20 08:00:00期间的集群状态, 特别是Weblogic集群、Nginx集群、ES集群、Kafka集群、Redis集群和Oracle集群。

MonitorAgent: 总结后的结果为: 所有检测项目均未发现异常信息。

RCAAgent: @TraceAgent, 请检测调用链相关信息, 定位是否有问题。

TraceAgent: 总结后的结果为: 在访问过程中出现故障, 占比38.49%, 同时伴有少量数据库写入错误, 违反数据库唯一约束。

RCAAgent: @TradeAgent, 检测交易相关指标出现问题时, 需要排查交易大量失败和交易耗时长的的问题。@LogAgent, 需要检测gc日志和access日志是否存在异常。

TradeAgent: 在该时间段内, 总体交易量同比下降, 相较于前一天下降了5.39%。

RCAAgent: @LogAgent, 需要检测该时间段内的access日志, 查看是否有异常交易量的情况。

LogAgent: 总结后的结果为: 订单集群的gc日志存在异常, 异常集群为Weblogic\_31, 主要表现为gc日志的GC数量异常。

CMDBAgent: 定位到方法或者集群/gc/, /, Weblogic\_31值得进一步检查

RCAAgent: 检索到以下3条专家经验/历史case可能相符, 结果如下:

专家经验: javaGC, 表现: JAVA进程GC异常, GC overhead limit exceeded错误, gc日志异常, 频繁GC, 措施OptimizeJVMSettings

专家经验: Traffic, 表现: 入网集群nginx等流量相关指标突增,调用链存在丢失点, 措施LimitTraffic

专家经验: CacheOverflow, 表现: Redis集群缓存占用过高, 导致新的写入请求失败, 措施IncreaseCacheSize

RCAAgent: 参考历史事故和专家经验, 推断问题根因为CacheOverflow, 通常的表现为Redis集群缓存占用过高, 导致新的写入请求失败, 建议采取IncreaseCacheSize措施



2023 CCF国际AIOps挑战赛决赛暨“大模型时代的AIOps”研讨会

# THANKS



2023 CCF国际AIOps挑战赛决赛暨“大模型时代的AIOps”研讨会

# THANKS