QCon
全球软件开发大会

InfoQ 极客传媒

突破泛化瓶颈：阿里云智能运维Agent评测体系实践

李也

阿里云 算法专家

目录

InfoQ 极客传媒

北京

QCon 全球软件开发大会

会议时间：4月10-12日

- 大模型赋能 AIOps
- 越挫越勇的大前端
- 云上业务架构演进
- 多模态大模型及应用
- 海外 AI 应用创新实践

北京

AiCon 全球人工智能开发与应用大会

会议时间：6月27-28日

- 端侧智能
- AI Agent
- 多模态大模型
- 金融+大模型

上海

QCon 全球软件开发大会

会议时间：10月23-25日

- AI Agent
- 大模型训练推理
- 端侧 AI
- 搜推深度融合
- 数智金融

4月 — 5月 — 6月 — 8月 — 10月 — 12月

上海

AiCon 全球人工智能开发与应用大会

会议时间：5月23-24日

- 通用大模型
- AI Agent
- 垂直领域应用
- 模型可解释性

深圳

AiCon 全球人工智能开发与应用大会

会议时间：8月22-23日

- 模型效率与部署
- 多模态大模型
- 大模型安全
- 智能硬件

北京

AiCon 全球人工智能开发与应用大会

会议时间：12月19-20日

- 通用大模型
- 智能硬件
- LMOps
- 具身智能

极客邦科技 2025 年会议规划

促进软件开发及相关领域知识与创新的传播

参会咨询

查看会议

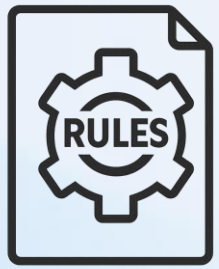
01

智能运维泛化之痛

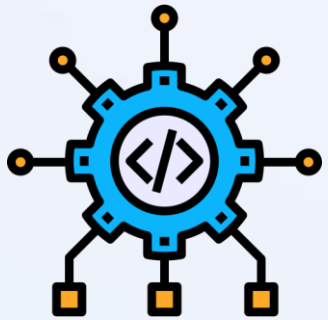
智能运维的一些技术

这些技术都有难泛化的问题

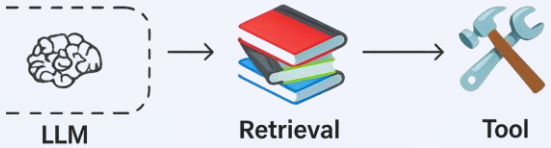
基于规则的智能运维



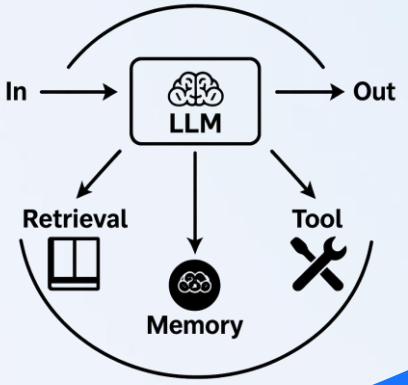
规则+算法的智能运维



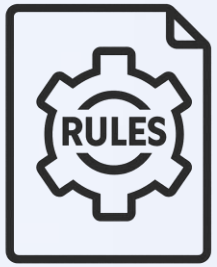
context engineering
+ 大模型 workflow



大模型智能运维Agent



传统的基于规则的智能运维难泛化



为CPU/内存/磁盘/响应时间等指标设置固定上/下限，超限即告警

泛化瓶颈：

- 环境变化：如大促
- 跨系统迁移差：不同应用/硬件

固定阈值的时序告警

用正则模板提取日志字段，基于关键字或模板频次设置规则

泛化瓶颈：

- 模板脆弱：日志格式/字段轻微变更就失效
- 难捕获未知错误

关键词/正则 匹配的日志告警

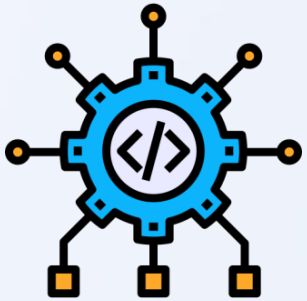
用“if-then”把事件聚合，如“网络断+下游告警→归为网络故障”

泛化瓶颈：

- 系统变化会使规则失效
- 组合爆炸：要处理各种事件的组合，规则数量会激增

基于If-else 规则的事件关联

规则+算法的智能运维难泛化



用STL分解/n-sigma/ESD-test等算法设置智能基线，超限告警

泛化瓶颈：

- 参数难设置：如敏感度，周期长度等

基线/季节性建模的时序告警

用spell/drain等算法挖掘出日志模板。对模板和变量的模式做告警

泛化瓶颈：

- 过度挖掘或者挖掘不充分
- 系统变化，概念漂移
- 过滤不充分: 无关日志造成误报

基于日志模板挖掘的日志告警

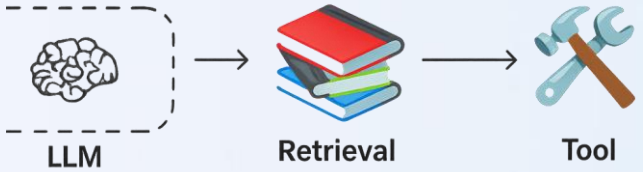
算法自动挖掘出if-then规则。随机游走/matrix-forest归因

泛化瓶颈：

- 规则挖掘依赖手动标注数据
- 数学模型的假设不一定成立
- 可观测数据不完整/有噪音

规则挖掘，随机游走等算法归因

大模型 workflow 会遇到泛化瓶颈



- 数据幻觉: 编造不存在的“500”
- 因果幻觉: 见GC增多就断言“内存泄漏”，忽略了Java版本变化
- 工具幻觉: 伪造API或者参数

模型幻觉

workflow的限定容易过强

- 从指标触发的 workflow，难泛化到日志报错的场景
- 3层的workflow难处理超过3步的根因推理
- workflow分支过多时，节点数爆炸

Workflow编排的局限性

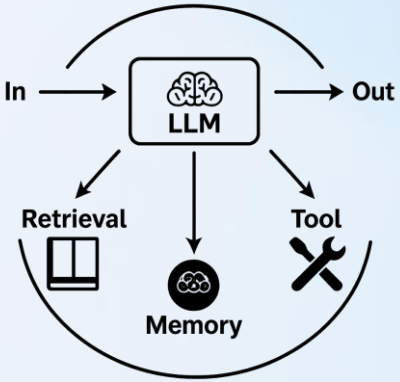
相比于确定编码的规则，大模型的输出可能会不遵从预期

- prompt说“只做只读分析”，模型也有可能执行危险命令（如直接重启集群、删库清表）
- 顶尖的模型在超过200个指令之后，也难全遵从[1]

不遵从提示词

[1] How Many Instructions Can LLMs Follow at Once? *Daniel Jaroslawicz et. al.*

大模型 Agent 也有一些泛化问题



Demo相对容易：接入开源的脚手架，准备好prompt和tools，就能在特定场景完成demo

大模型Agent继承了大模型workflow里面的很多缺点

- 数据/因果/工具幻觉
- 不遵从提示词

还新增了更多执行流程上的缺点

- JSON、SQL等易出现格式和类型错，影响自动化
- 难以停止，重复探索

大模型能力局限

- 工具不全
- 工具太多
- 工具作用重叠
- 工具描述不清

工具不成体系

“按下葫芦起了瓢”

- 强调“仅基于证据回答”，模型过度保守，提早终止
- 强调“尽量只用内部知识库”时，模型不做合理泛化
- 强调“严格特定schema 格式”，模型为过度满足schema而填充mock值

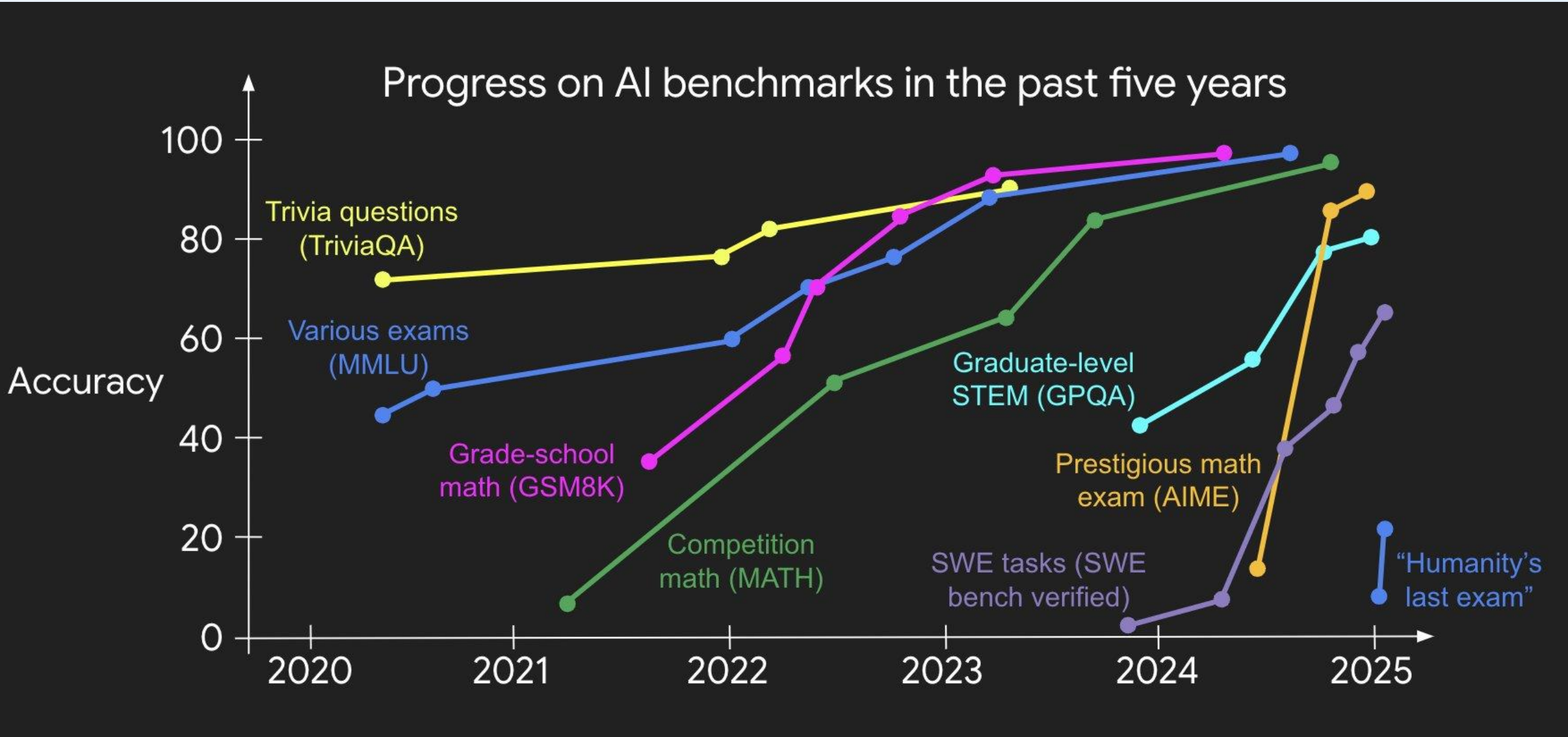
调prompt没有掌控感

02

高质量的评测集的重要性

各个领域的评测集(benchmark)推动了大模型在这些领域的进步

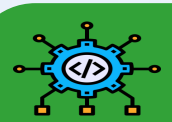
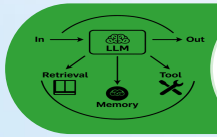
https://ysymyth.github.io/The-Second-Half/



如果我们用一个案例的Demo做评测

例子: 我们想要做一个下面场景的的智能运维的Demo

现象是出现了很多慢SQL，原因是有一次代码变更使用了长连接，把数据库的线程池打满了。
我们要把“代码变更”这个原因找到

	**规则**	“超过1s的是慢SQL”，“active_session数量超过CPU核数两倍则认为是打满” “同时发生 代码变更+线程池打满+慢SQL 时认为代码变更是原因”
	**规则＋算法**	“将SQL的执行时间用STL分解后，残差超过均值+3倍标准差是慢SQL”，…， “在因果图里面计算出代码变更是根因的概率”
	**大模型workflow**	“先调用工具对感兴趣的指标和日志做一遍巡检”→“对于有异常的现象用大模型进行总结并输出”→“将上面对根因的判断的规则写到prompt里面然后做根因定位”
	**大模型Agent**	“慢SQL检测工具+线程池检测工具+代码变更识别工具”，“案例retrieval知识库”，然后把大模型workflow里面硬编码的流程写到prompt里面

我们可以想象，用各种方法我们都能在上面的案例上做一个成功的demo。但是可能难泛化到其他场景

如果有一个更丰富benchmark

把前面提到过的例子都作为benchmark的任务，去评测上面各种方法的Demo						
方法\任务	变更导致的数据库连接池打满	受季节变化和大促影响的CPU使用率异常检测	Java版本变化导致的GC次数上涨	网络断+下游多告警	…	特定流量造成的报错量上涨
[Demo] 规则	✅	❌	❌	❌		❌
[Demo] 规则+算法	✅	✅	❌	❌		❌
[Demo] 大模型workflow	✅	❓	❓	❓		❓
[Demo] 大模型agent	✅	❓	❓	❓		❓

上面的❓表示有一定的概率可以做对

“高质量”的benchmark能帮我们发现AIOps方法的局限性

我们会在下一节讨论什么是“高质量”。我们先把“高质量”理解成覆盖我们常见的方方面面问题的案例集，而且还附有标准答案和评判标准

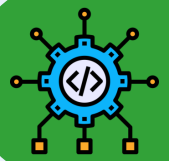
一个可能的“高质量”的benchmark							
案例	变更导致的数据库连接池打满	受季节和大促影响的CPU使用率异常检测	Java版本变化导致的GC次数上涨	网络断+下游多处告警	…	…	…
标准答案	代码变更	有异常/无异常	Java版本升级	A→B 网络断开	…	…	…
可观测数据	系统在正常和异常时间段的所有logging/tracing/metrics/events数据，至少够人类专家找到答案						

- **发现方法的局限性**
- **排除不可行的方法**
- **提前暴露问题**

“高质量”的评测集能帮我们优化AIOps方法

规则

不断堆规则，直到能覆盖90%以上的benchmark里面的问题

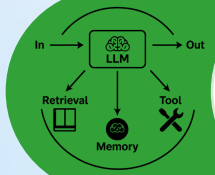


规则＋算法

根据benchmark的结果，修改算法参数。根据benchmark，自动挖掘规则

大模型workflow

根据benchmark的结果调整workflow的结构和prompts，减少bad case



大模型Agent

设计更巧妙的脚手架 + 开发更多好用的工具 + 优化prompts
使用标注数据去做 监督微调/强化微调/Agentic强化学习

benchmark促进循环迭代

开发

开发的时候可以用benchmark中的少量案例来验证代码是否可以运行

评估

初步开发完之后，我们可以在全量benchmark上评估算法的效果

开源

将能共享的benchmark数据都尽量共享，借助社区的力量

上线 + 积累更多案例

上线后会产生用户真实的反馈。这些反馈能帮我们积累更多案例，又可以加入到benchmark中形成循环

调优

对于bad cases，我们可以拿出来分析然后有针对性地调优算法

03

如何构建高质量的评测集

我们对软件系统执行要素的抽象以及对应的注入点

软件系统包括了下面的这些方面，可能的根因可能和上述某个方面/多个方面有关

系统请求 workload

系统处理的请求明细及其处理时间，代表了系统的输入

对应注入点: 整体流量突增/突降，特定入参的请求突增 等

系统资源 resources

例如 CPU/内存/线程池 等

对应注入点: 内存不足、CPU不足、线程池打满 等

代码/config

软件/程序 及其配置。包括了代码的版本等

对应注入点: 代码变更、特定输入触发的bug、参数欠佳 等

系统历史状态

数据库/缓存/消息队列的历史状态，内存快照等

对应注入点: 复制延迟、redis大key、DB统计信息过期 等

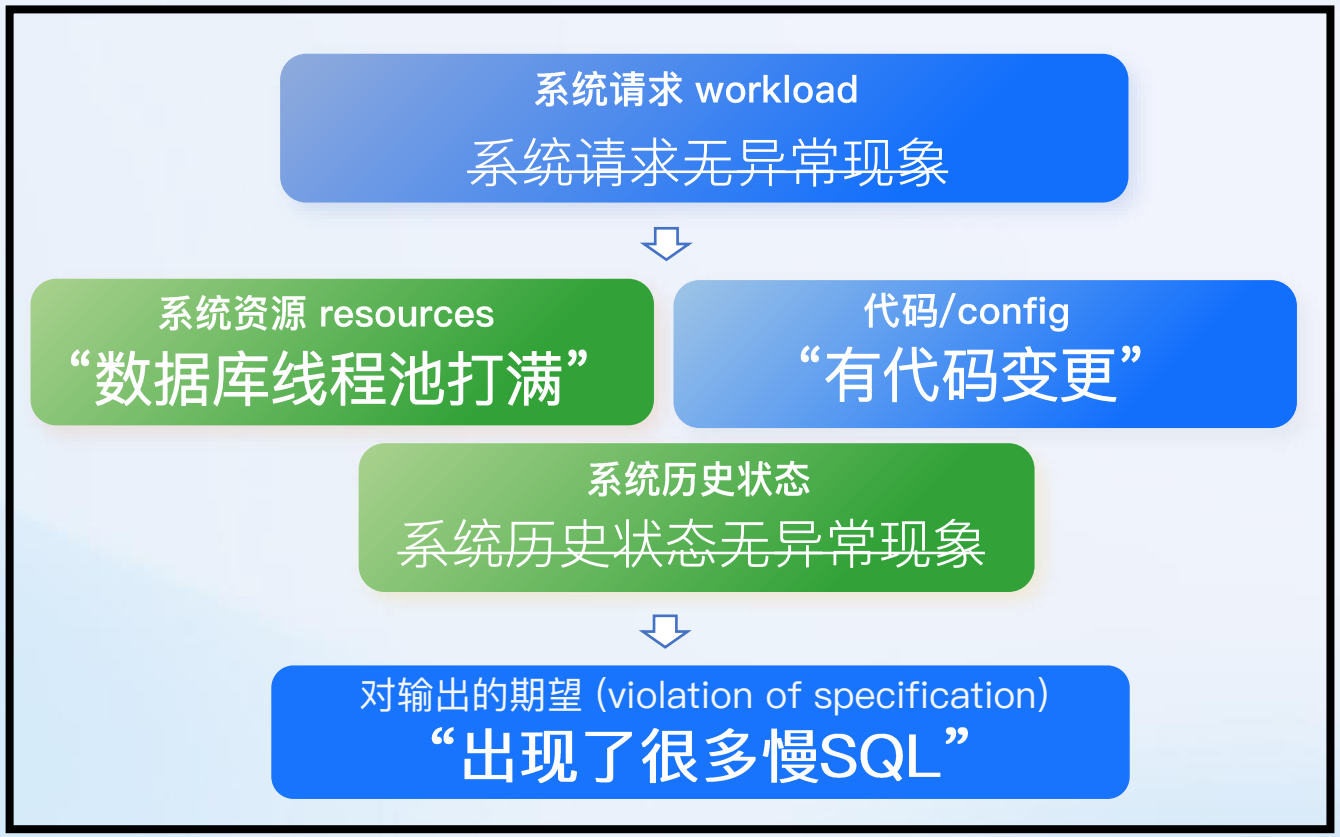
对输出的期望（specification）

这部分对应的是注入的故障造成的影响，没有注入点

例子

例子: 我们想要做一个下面场景的的智能运维的Demo

现象是出现了很多慢SQL，原因是有一次代码变更使用了长连接，把数据库的线程池打满了。
我们要把“代码变更”这个原因找到



- 对每个案例，我们都能按照这样的方式将可观测数据和前因后果组织起来，形成很多 根因/现象/结果 等类型节点
- 如果我们的benchmark覆盖了任何一个工单中出现过的根因/现象/结果，那么我们就说覆盖度是100%

高质量benchmark的几个要点

覆盖度是指要尽量各种类型的原因/现象/影响

覆盖度

覆盖根因

覆盖我们能想到的和系统各个要素相关的根因：
如“流量暴涨”，“CPU利用率太高”，“内存打满”等

覆盖传播链条

覆盖我们能想到的影响传播链路：
如“流量暴涨”→“数据库CPU打满”→“请求延迟高”
如“网络配置变更”→“控制平面队列积压”→“网络丢包增多”

覆盖结果

覆盖我们能想到的最终影响的结果，包括“请求延迟高”/“报错数增多”等

真实度是要让系统产生的可观测数据尽可能接近真实

真实度

系统架构真实

需要产生可观测数据的系统架构接近真实的架构，包括了真实的组件等

流量真实

需要能模拟真实用户的流量

各组件的可观测数据真实

日志要像真实的日志，时序要像真实的时序。真实的可观测数据在正常时段会有噪音，在异常时段会有同时出现的unknowns

案例来源

基于生产环境中真实发生的故障样本进行归纳与复盘
- 真实可观测数据的快照
- 专家标注

真实故障收集

用混沌工程工具
以及模拟workload
在开源真实系统 (如otel-demo) 中触发典型故障
- 实采的可观测数据
- 注入故障作为标注

开源系统故障注入

在与生产相近的预发/演练环境中注入故障
- 实采的可观测数据
- 注入故障作为标注

演练环境故障注入

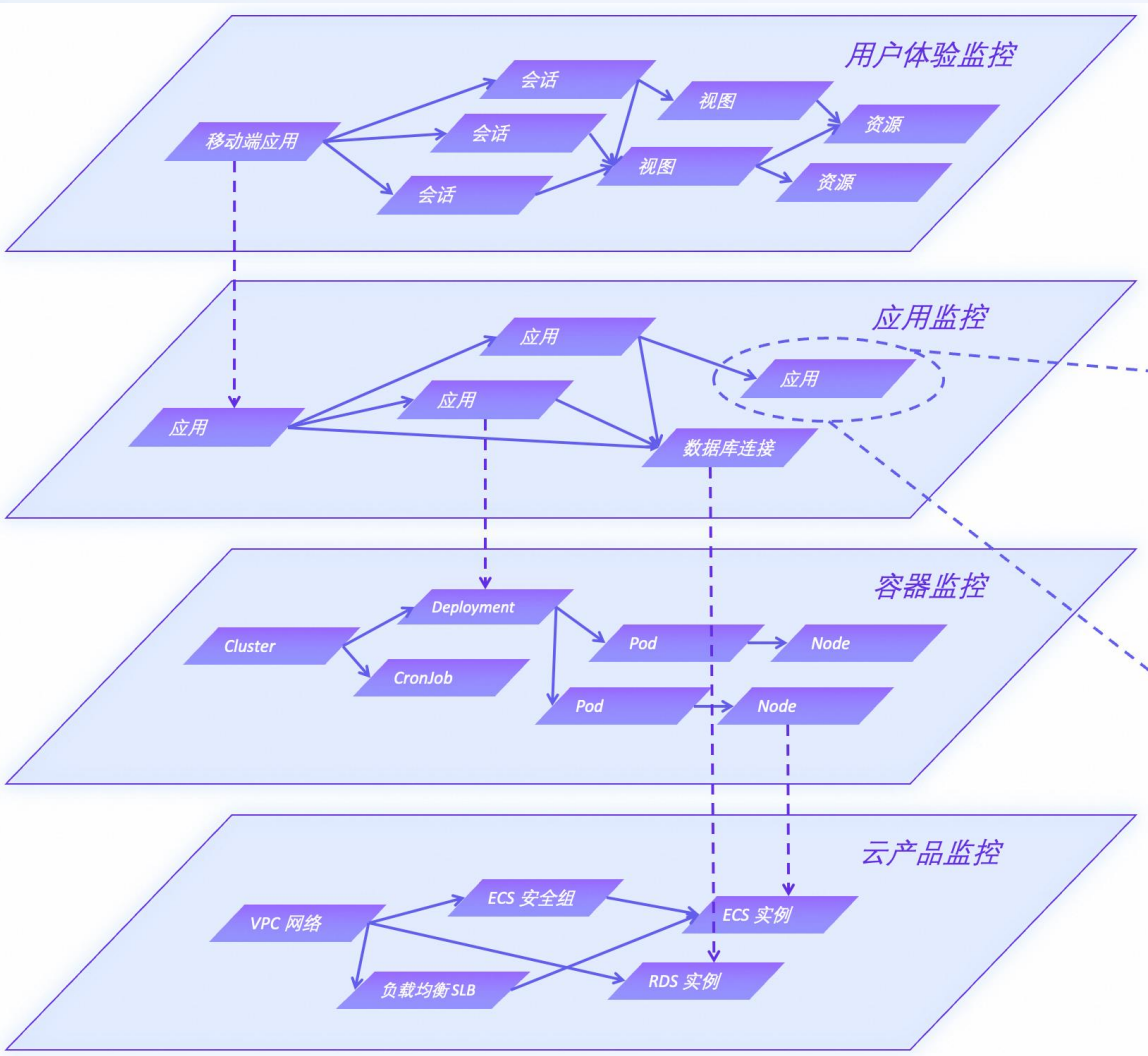
用混沌工程工具
以及模拟workload
在mock了函数输入输出的真实运行系统中触发典型故障
- 实采的可观测数据
- 注入故障作为标注

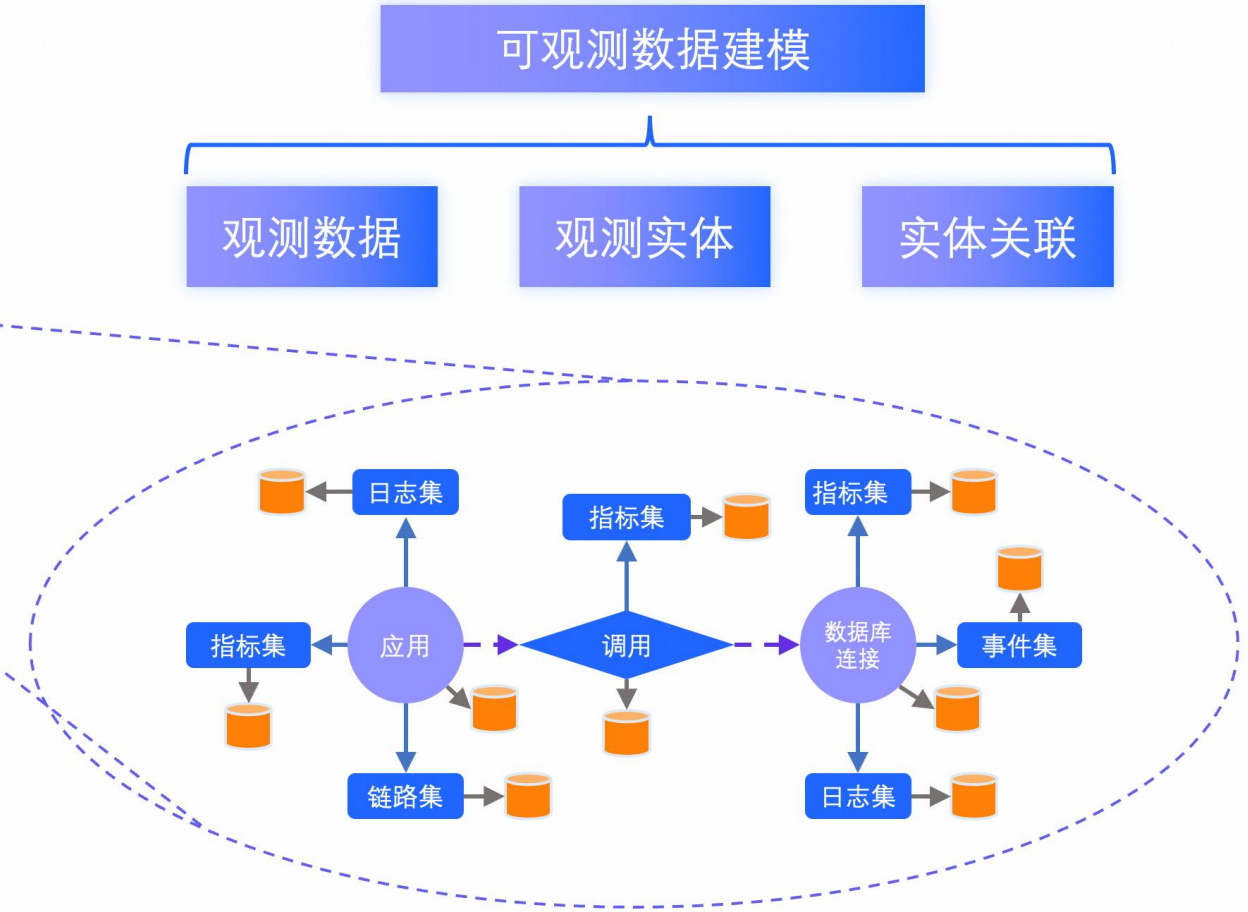
模拟系统故障注入

不同来源的案例在几个维度上的区别

注入方式	覆盖度	真实度	可公开
真实 case 收集	★★ (容易缺可观测数据/标注)	★★★★★	★
开源系统故障注入	★★★ (需要投入大量的时间精力)	★★ (开源系统的架构和生产有区别)	★★★★★
演练环境故障注入	★★★ (需要投入大量的时间精力)	★★★★ (workload难复现)	★★ (包含了真实的组件)
模拟系统故障注入（Mock/仿真）	★★★★ (模拟的成本低，易覆盖)	★★★ (workload难复现)	★★★★★

统一模型（UModel）





可观测 UModel 模型定义

从理论角度提升覆盖度 方法论：统一模型umodel下等价

例子: 我们想要做一个下面场景的的智能运维的Demo

现象是出现了很多慢SQL，原因是有一次代码变更使用了长连接，把数据库的线程池打满了。
我们要把“代码变更”这个原因找到

用umodel来表达案例的前因后果

- **等价性**: 任何案例都可以用 umodel 来表达前因后果关系。如果两个案例用umodel写出来形式一致，那么我们说这他们在umodel下是等价的
- **完备性**：如果我们能想到的所有案例都能在benchmark中找到某个案例在umodel下等价。那么我们说benchmark在umodel下是完备的
- **必要性**: umodel里面所有的实体类型和指标集合都必须要出现在benchmark中的某个案例里面

04

阿里云AIOps评测集(持续发布)

阿里云AIOps评测集的构建

实施注入/案例收集	已注入2000+ cases	经验证已收集 500+ cases	持续收集中			
故障注入系统/规划	chaosblade	chaosmesh	系统请求workload类	系统资源resource类	代码/config类	系统历史状态类
可观测数据采集+存储+查询分析	阿里云 云监控2.0	阿里云 日志存储SLS/指标存储SLS MetricStore				
接入系统	OpenTelemetry demo开源系统	阿里云内部演练环境	真实生产系统	模拟系统	…	

可以和其他benchmark互补使用，如OpenRCA

实施注入/案例收集	335 cases					
故障注入系统/规划	~~chaosblade~~	~~chaosmesh~~	~~系统请求workload类~~	系统资源resource类	~~代码/config类~~	~~系统历史状态类~~
可观测数据采集+存储+查询分析	csv files, 68GB	~~用SQL/SPL查询，开箱即用的可视化~~				
接入系统	Telecom database	Bank system	Market system			

OpenRCA的数据简介

第一批评测集 发布

第一批评测集随AI原生编程挑战赛一同发布
所有人可以通过自己的阿里云账号去云监控2.0 / SLS里面查询数据

后续会陆续开源新的评测集，欢迎关注“阿里云云原生”公众号

[2] https://tianchi.aliyun.com/specials/promotion/2025aicloudnativecompetition

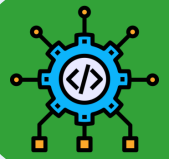
05

基于评测集的智能运维 Agent 能力提升实践

“高质量”的评测集能帮我们优化AIOps方法 — 回顾

规则

不断堆规则，直到能覆盖90%以上的benchmark里面的问题

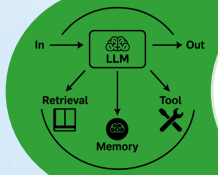


规则＋算法

根据benchmark，修改算法参数。根据benchmark，自动挖掘规则

大模型workflow

根据benchmark的结果调整workflow的结构和prompts，减少bad case



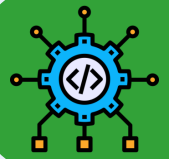
大模型Agent

设计更巧妙的脚手架 + 开发更多好用的工具 + 优化prompts
使用标注数据去做 监督微调/强化微调/Agentic强化学习

一些根据benchmark调优的结果

规则

纯规则的方案就暂不调优了

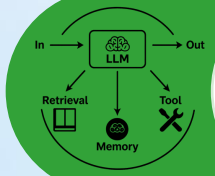


规则＋算法

3000+ 样本 自动挖掘规则的算法达到了 100%准确率 和 99.27%的召回率

大模型workflow

2000+ 样本 正则表达式生成的benchmark。生成的正则表达式中98.48%的能匹配文本，95.6%的正则可以完美解析80%以上的字段



大模型Agent

精调过的脚手架 + 工具 在第一批评测集上达到了87.5%的根因召回率
经过 监督微调/强化微调 的模型在RCA根因排序任务上达到了82.0%的准确率

InfoQ 极客传媒

北京

QCon 全球软件开发大会

会议时间：4月10-12日

- 大模型赋能 AIOps
- 越挫越勇的大前端
- 云上业务架构演进
- 多模态大模型及应用
- 海外 AI 应用创新实践

北京

AiCon 全球人工智能开发与应用大会

会议时间：6月27-28日

- 端侧智能
- AI Agent
- 多模态大模型
- 金融+大模型

上海

QCon 全球软件开发大会

会议时间：10月23-25日

- AI Agent
- 大模型训练推理
- 端侧 AI
- 搜推深度融合
- 数智金融

4月 — 5月 — 6月 — 8月 — 10月 — 12月

上海

AiCon 全球人工智能开发与应用大会

会议时间：5月23-24日

- 通用大模型
- AI Agent
- 垂直领域应用
- 模型可解释性

深圳

AiCon 全球人工智能开发与应用大会

会议时间：8月22-23日

- 模型效率与部署
- 多模态大模型
- 大模型安全
- 智能硬件

北京

AiCon 全球人工智能开发与应用大会

会议时间：12月19-20日

- 通用大模型
- 智能硬件
- LMOps
- 具身智能

极客邦科技 2025 年会议规划

促进软件开发及相关领域知识与创新的传播

参会咨询

查看会议

QCon
全球软件开发大会

InfoQ 极客传媒

THANKS

大模型正在重新定义软件

Large Language Model Is Redefining The Software