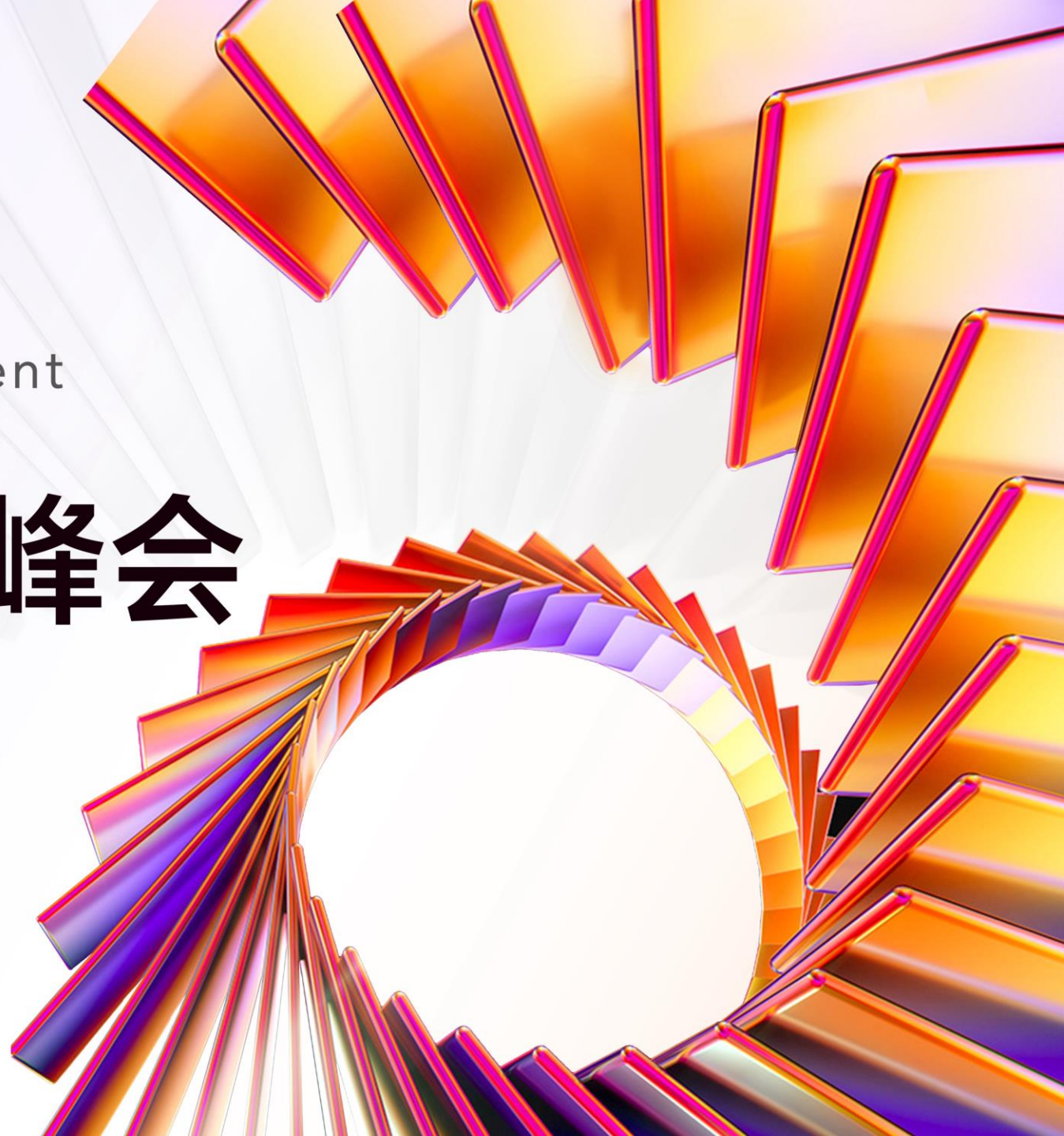
AiDD 7th 2025

第7届 AI+ Development Digital Summit AI+研发数字峰会

拥抱AI 重塑研发

8月8-9日 | 北京站



AiDD 8th 2025

第8届AI+研发数字峰会

拥抱AI 重塑研发 AI+ Development Digital Summit

查看会议详情

下一站预告 11/14-15 | 深圳站 12/19-20 | 上海站

深圳站论坛设置

智能装备与机器人

超越“编程 Copilot”

下一代知识工程

智能网联与汽车智能化

AI 测试工具开发与应用

AI 基础设施和运维

数据智能及其行业应用

可信 AI 安全工程

大模型和 AI 应用评测

多 Agent 协同框架

从智能测试到自主测试

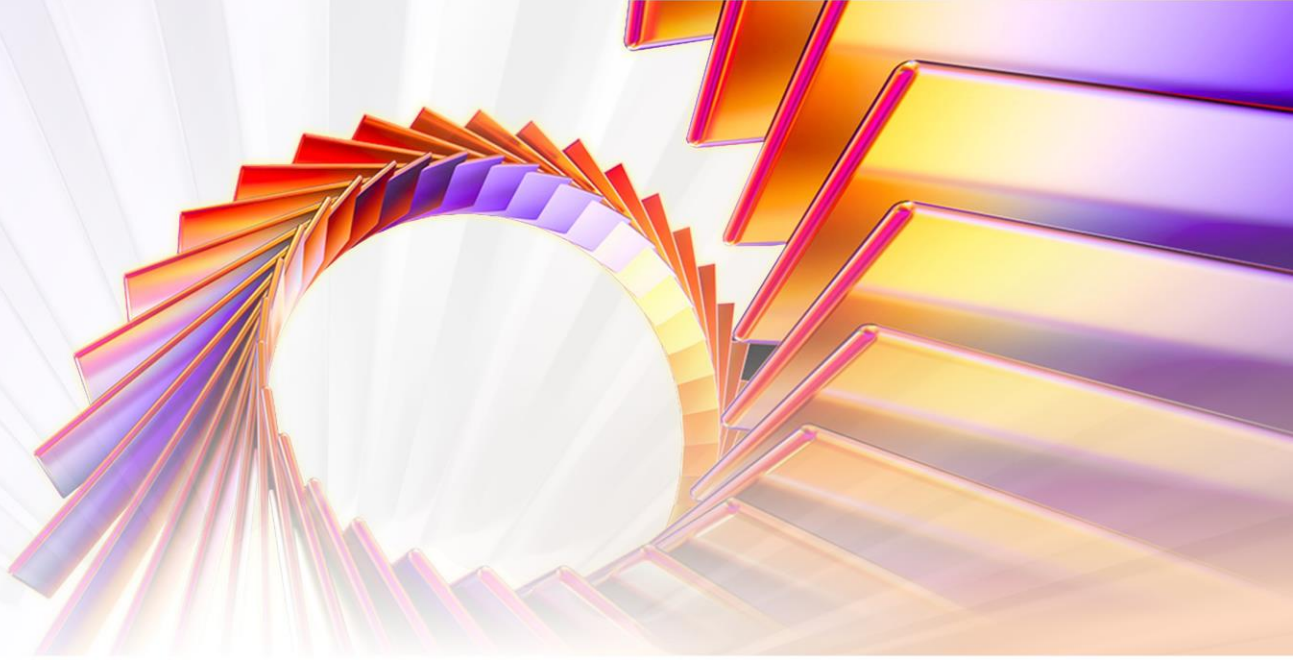
大模型推理优化

多模态 LLM 训练与应用

智能化 DevOps 流水线

上下文工程

AiDD 7th 2025 | 8月8-9日 | 北京站
第7届 AI+ Development Digital Summit
AI+研发数字峰会
拥抱AI 重塑研发



Meta-Agent架构与业务测试落地实践

杨青霖 | 百度

杨青霖

百度资深测试开发工程师

主要负责语音终端和视觉算法的质量保证体系建设及工程效能提升。拥有将Python应用于复杂技术领域的丰富实战经验。著有《Python气象应用编程》，人民邮电出版社异步社区2023年度影响力作者。

现在负责语音业务测试工具研发，带领团队探索AI在测试领域的应用。每天与各种语音测试场景打交道。

目录
CONTENTS

孟祥银

[SpeechSolution-30180] 【雨燕】识别解码器-支持通过flush次数控制长音频计算频次-提测

iCafe卡片

【雨燕】识别解码器-支持通过flush次数控...

类型：Story

状态：待测试

负责人：杨娟娟,迪力亚尔·帕尔哈提

所属计划：未计划

你是卡片的负责人、QA负责人、创建人、最后...

如流工作卡

孟祥银

提测了

迪力亚尔·帕尔哈提

引用 孟祥银：

https://console.cloud.baidu-int.com/devops/icafe/issue/SpeechSolution-30180/show

@阿童木 机器人 准入

输入消息

PART 01

背景与演进方向

智能体的现状与困境

语音测试

- 提测
- 准入
- 准入
- 功能测试
- 声学测试
- 稳定性测试

理想的AI测试助手

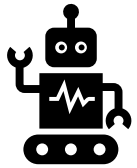
应该是什么样的?

尴尬的等待

执行一下稳定性测试

好的，正在执行中。

执行中...



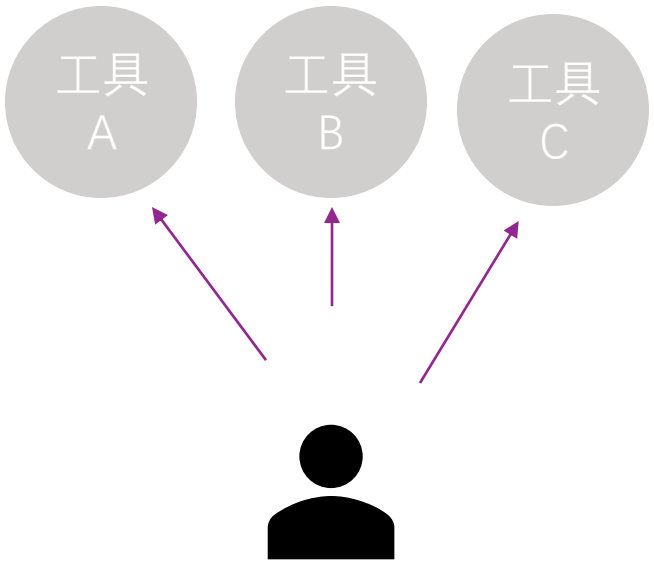
你还在吗？

我们满怀期望地引入Agent，却发现...

当它处理一个耗时任务时，整个对话就被阻塞了。

三种范式演进

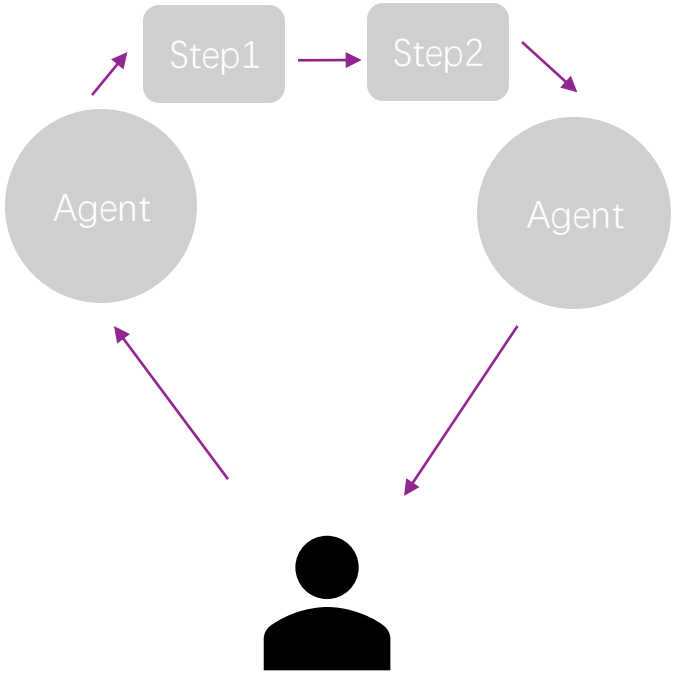
工具箱模式



体验割裂，上下文不互通

像一堆独立的计算器

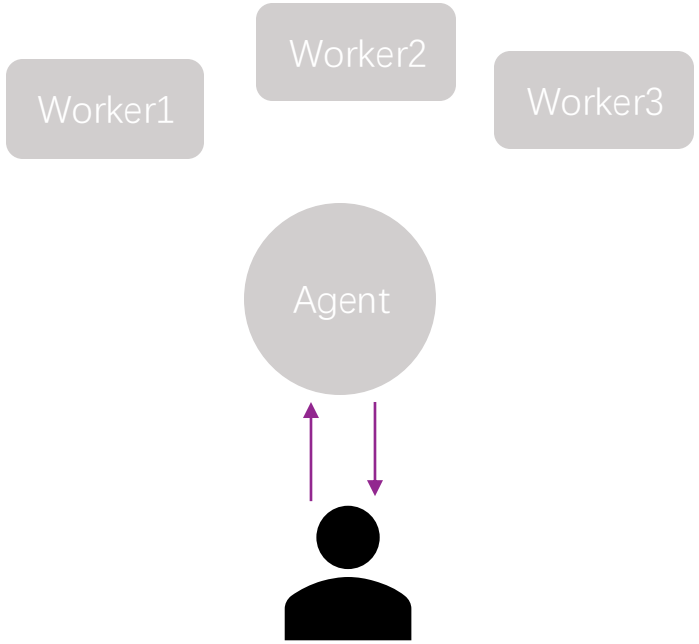
流水线模式



强大但僵化，用户必须等待响应

像一条流水线

项目经理模式



从“串行指令”到“并行委托”

像一个项目经理

项目经理模式

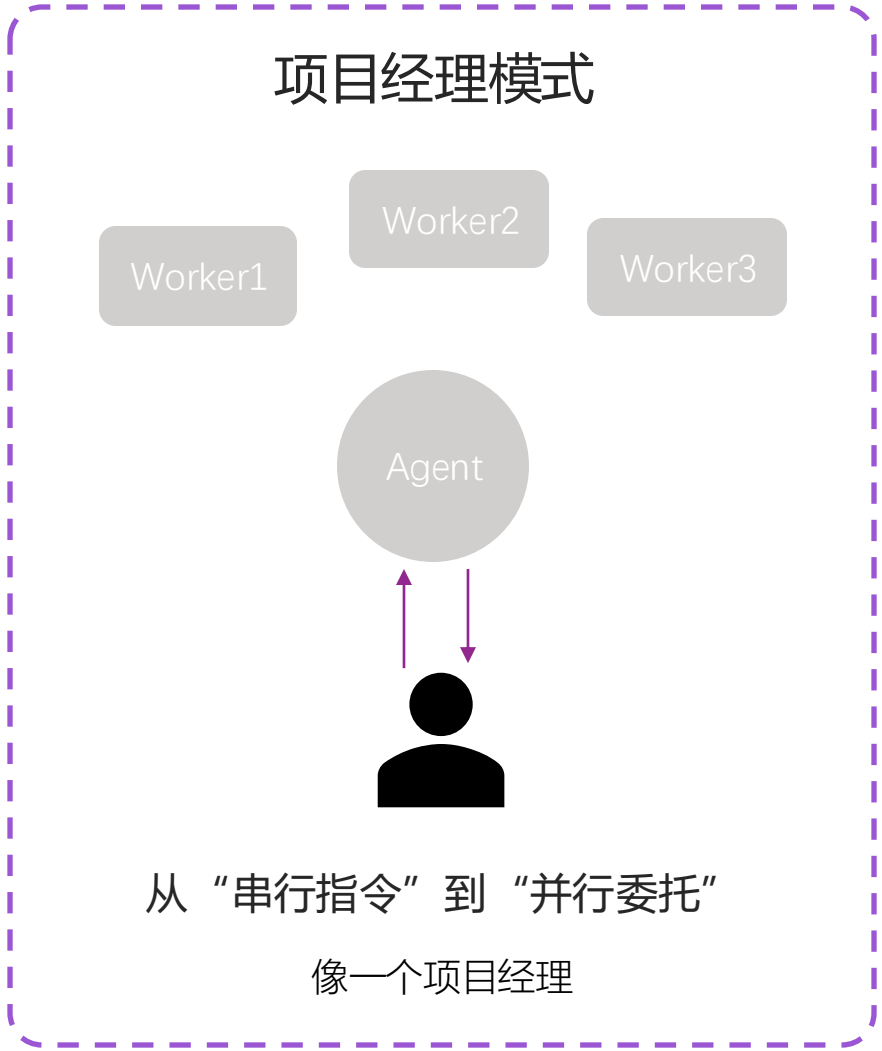
以企业IM入口为例

企业IM的特点 （机器人、服务号）

- 单线程：同一个消息发送对象，通常只能开一个对话窗口；
- 无明显消息边界：消息之间难以找到明确的对话分界线。

需要解决的问题

- 在单线程对话下执行多个任务；
- 长耗时任务且不阻塞对话流；
- 主动通知任务结果；
- 符合用户直觉的连续消息流。



PART 02

构建非阻塞核心

方案

单线程对话下执行多个任务，且长耗时任务不阻塞对话流

- 用户的任何消息应该尽快响应；
- 长耗时任务应该返回一个handler，而不是等待完成；
- 允许用户动态变更执行流程；
- 能从大量工具中选取需要的工具并规划流程；

异步操作

ReAct 调度器

工具粗选策略

主动通知任务结果，且消息流符合用户直觉

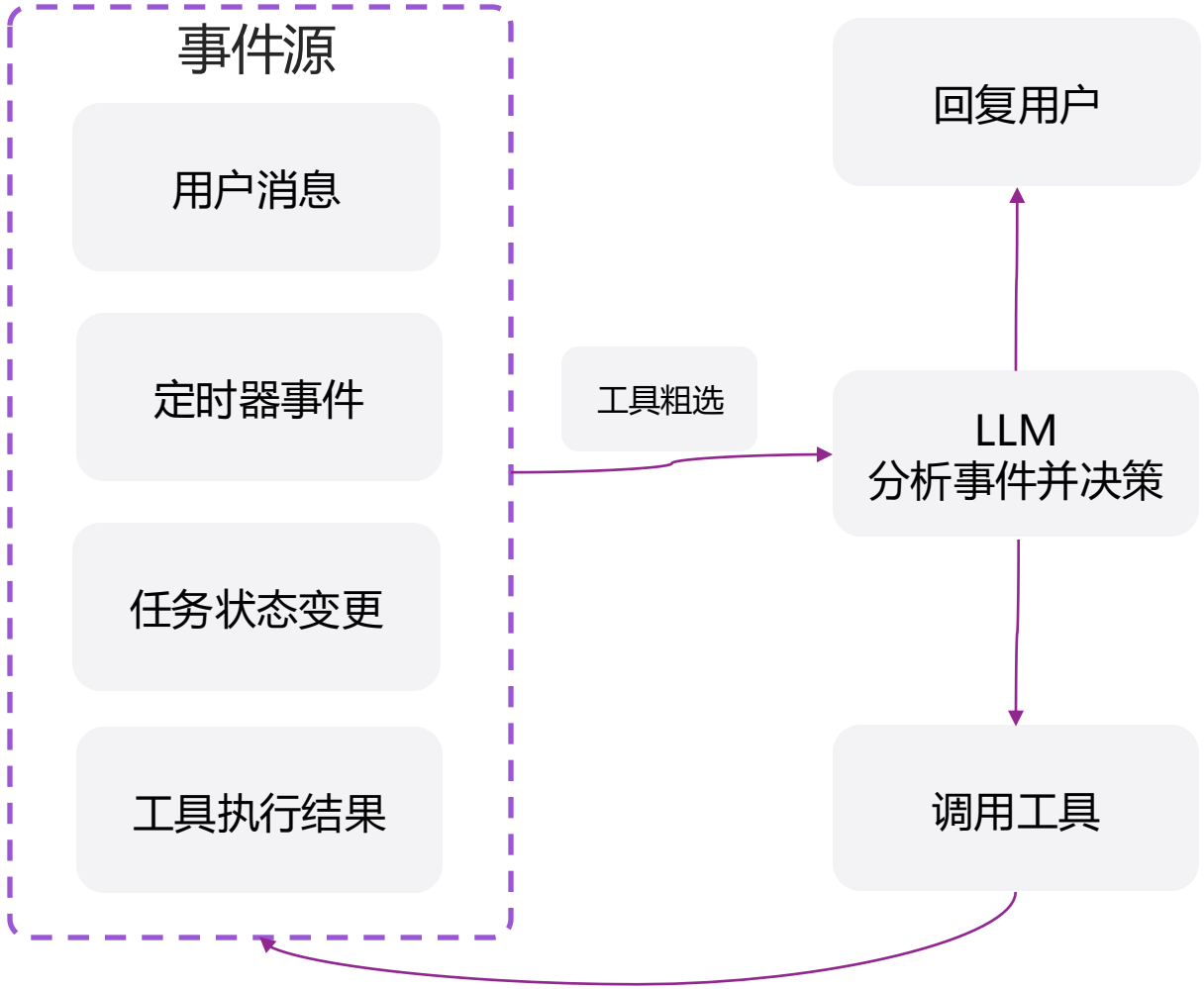
- 长耗时任务完成后，应该能触发下一步操作；
- agent应该知晓以自己身份所发出的任何一条消息；
- 在无边界的对话流中，没有明显的上下文中断感；

事件驱动

架构总览

IM回调 API	统一适配不同入口消息
调度器	异步、事件驱动的ReAct调度器
分层记忆系统 工具粗选策略	上下文系统与工具粗选
数据库 外部工具API 其他基础设施	运行支持

调度器 ReAct 架构



工具类型

同步工具

能立即获得结果的工具

直接返回结果

异步工具

需要消耗长时间运行的工具

返回任务handler

任务handler会被框架层持久化到数据库
并轮询状态更新

记忆并非越多越好

上下文窗口有限

LLM有固定的Token上下文长度，Agent不能无限提供所有记忆，否则会超出限制导致截断或拒答。

噪音与干扰

不相关或陈旧的信息会成为噪音，降低LLM判断的准确性。不加筛选地提供大量上下文，反而可能误导AI。

成本与效率

更长的上下文意味着更多的调用成本和时间延迟。每次对话塞入大段历史会显著增加API费用，并拖慢响应。

三层记忆体系

包括用户提问、助手回答、工具调用结果和系统提示等。超限就触发流转算法把最早部分摘要后移入中期记忆。

被动式记忆

短期记忆

- 对话内容
- 工具调用结果

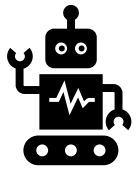
摘要提取

存入RAG数据库

中期记忆

- 近期对话摘要
- RAG召回

主动式记忆



主动更新

长期记忆

- 用户偏好
- 用户画像

中期记忆遗忘策略

模拟记忆曲线模拟：

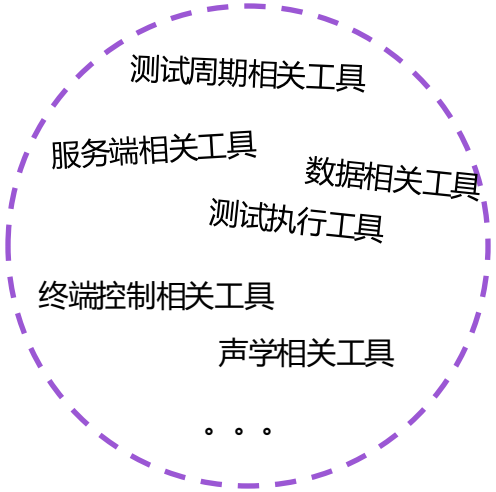
- 越新的记忆权重越高
- 常想起的回忆会更牢
- 太久以前的记忆会遗忘

为什么要设计遗忘策略？

只有进没有出的记忆可能是个灾难。

海量工具怎么办

可供调用的工具



100+

未来还可能继续增加

全量作为LLM的调用参数?

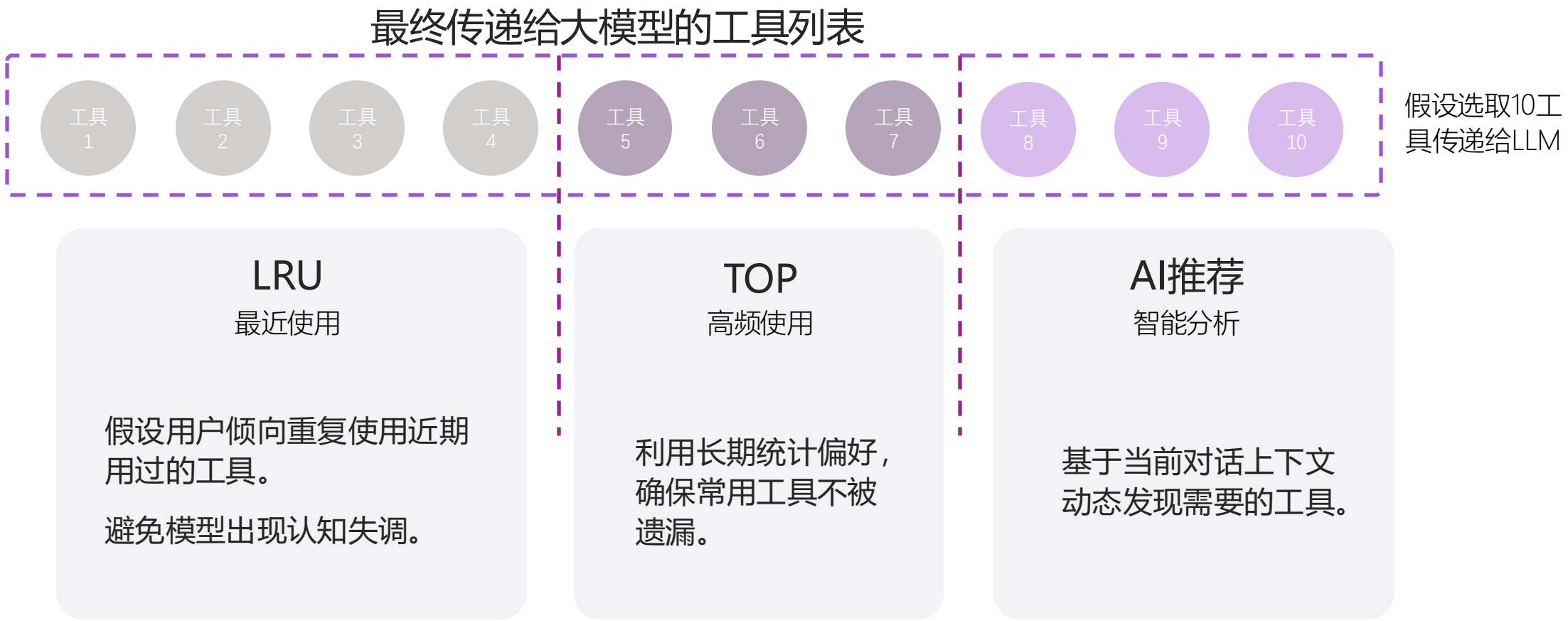
- 任务准确率降低
- 上下文截断
- 成本失控

选取子集， 怎么选?

- 按照集合让用户手动开启关闭?
- 根据用户消息RAG?

工具粗选的策略 – 槽位机制

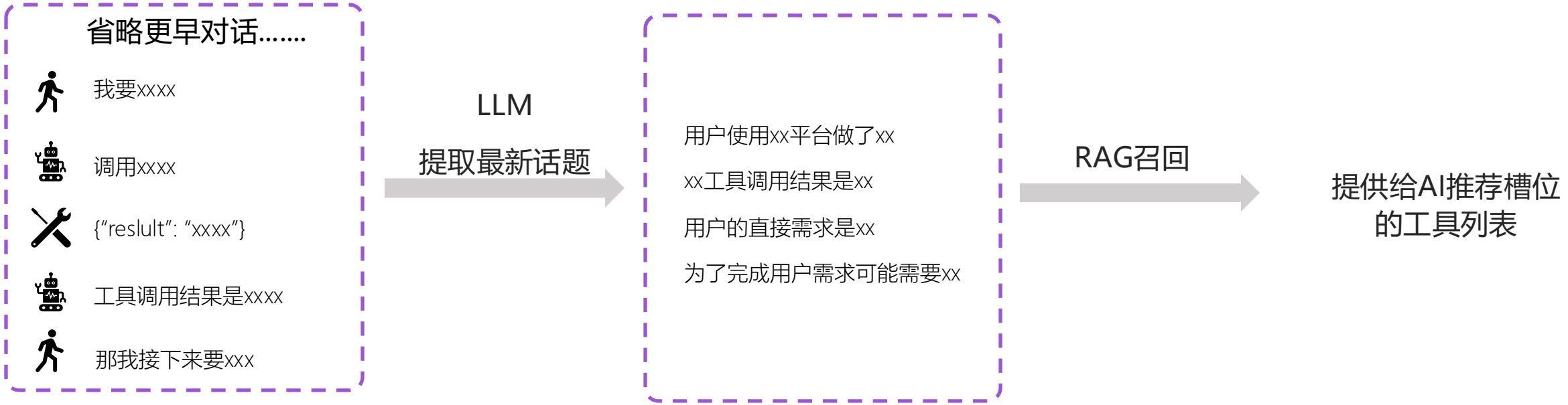
前面提到，每一次事件发生，都会引发一次工具粗选。



三个部分进行集合合并，如果已选工具数量小于总槽位大小，则以AI推荐的更多结果进行填充

AI推荐槽位

基于话题的推荐机制



上下文工程

直到目前，我们已经有“对话记忆”和“粗选工具集”，让我们把这些全部组合起来

请求 LLM 的Payload

详细对话

role: user/assistant/tool

短期记忆。

System Prompt

role: system

agent人设
基础规则

中期记忆

长期记忆

异步任务状态

定时任务状态

外部事件

用户信息

对话渠道

当前时间

工具列表

槽位机制粗选出来的工具列表，不再赘述。

轻量声明式的工具接入

我们发现，在能明确Agent应用范围的小团队中，共享代码库的声明式配置更为高效，
几分钟就能为Agent增加一项新技能。

```python
@pydantic.validate_call
async def my_custom_function(
    param1: typing.Annotated[str, pydantic.Field(description='参数描述')]
) -> str:
    """自定义工具函数"""
    # 实现你的业务逻辑
    return f"处理结果: {param1}"
```

工具声明

- 运行时类型校验；
- 明确参数类型；
- 无重复定义；

```python
async def init(atom_service: aeo.service.atom.AtomService, app: fastapi.FastAPI):
    """工具初始化"""
    atom_service.agent.add_tool_function(
        my_custom_function,
        aeo.helper.atom.types.FuncConfig(
            owner='your-name',
            tags=['custom'],  # 权限标签
            require_confirm='Optional'  # 可选: Required, Optional, Never
        )
    )
```

工具注册

- 配置负责人，以便出错时提示用户；
- 支持高风险工具运行前确认

工具的描述

怎样写一个工具的描述?

- 表达简洁通顺;
- 句式可以是 “输入什么，在什么平台做了什么，输出什么”；
- 可以举一个例子（one-shot），让LLM明白能做什么;
- 说明可能出现的异常情况，并给出解决建议;

小技巧

把工具描述给不同业务的同学看，如果他能明白这个工具在做什么，什么时候需要用，那这就是一个优秀的工具描述。

PART 03

业务落地

传统测试流程 & 痛点

需求分析 → 用例设计 → 测试执行 → 数据分析 → 报告生成

耗时费力

很多步骤纯人工完成，一个完整测试周期大量时间花在重复数据处理和结果汇整上。

易错缺乏一致性

人工设计用例和检查结果容易遗漏场景或计算出错，不同测试人员输出的质量参差不齐。

流程碎片化

各环节缺乏统一平台串联。需求-测试-交付流程割裂，难以量化覆盖度。

协作成本高

对接耗时。反馈周期长，问题定位延迟。测试报告以邮件形式分发，不易追溯管理。

如何重塑测试流程

服务端：

- 服务端功能测试
- 服务端算法测试
- 编写测试报告

终端：

- SDK功能测试
- SDK算法测试
- 编写测试报告

测试生命周期：

- 提测
- 准入
- 准出

测试过程：

- 故障排查
- BUG创建
- BUG验证/关闭

CI/CD平台控制：

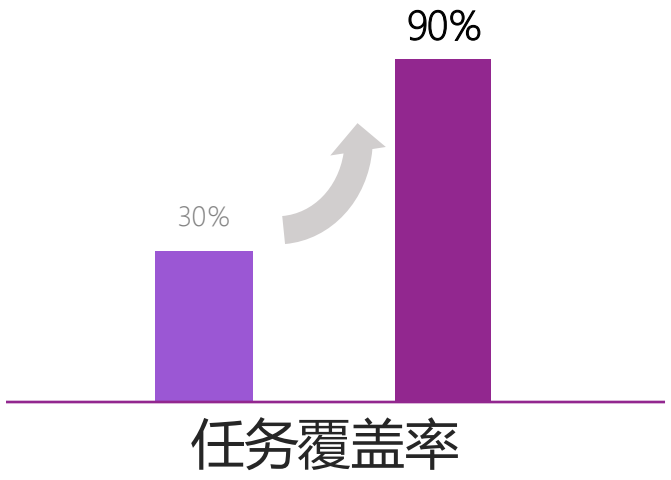
- 执行流水线
- 停止流水线
- 重新运行

数据管理：

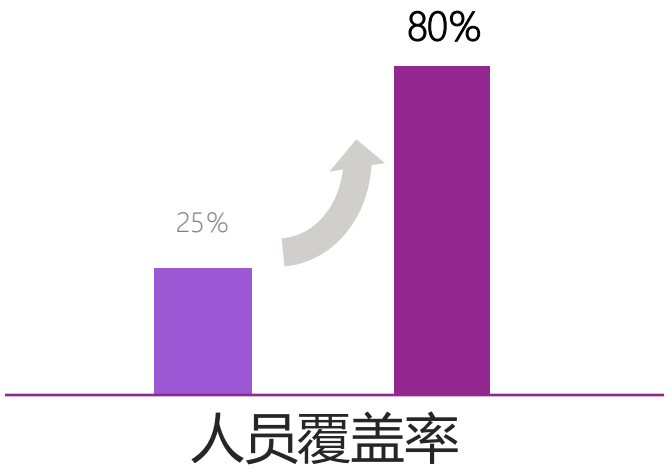
- 数据质检
- 数据预处理
- 数据管理

ALL IN ONE

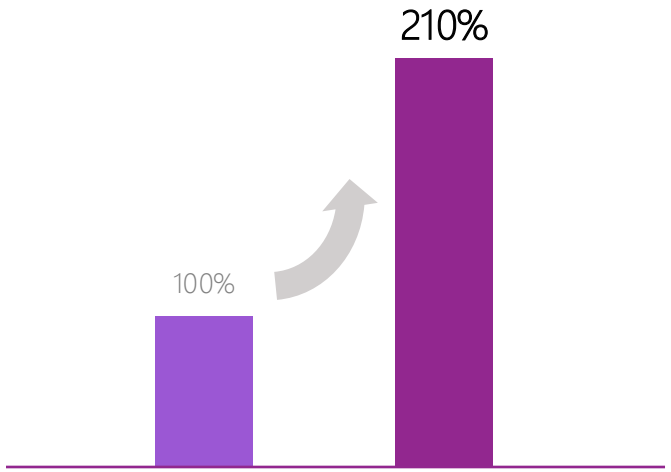
效率提升



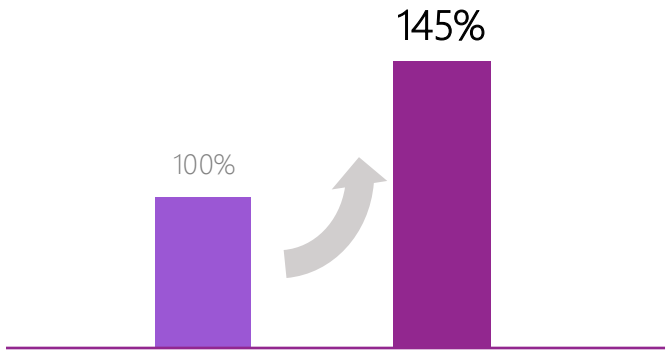
任务覆盖率



人员覆盖率



使用率



交付效率

什么是meta-agent

“Meta” 这个词根源于希腊语，意思是“之后”或“超越”(beyond)。

熟悉的例子：

- Metadata (元数据)：它不是数据本身，而是关于数据的数据。服务端算法测试
- Metacognition (元认知)：它不是指思考具体问题，而是指对思考的思考。

一个普通的 Agent，它的关注点是任务。你让它写代码，它就去写代码；你让它查资料，它就去查资料。它是一个执行者。

Meta-Agent 正是普通 Agent 的一个“Meta”层。它的关注点是流程和 Agent 本身，是一个负责协调、监督和管理其他 Agent 的 “元”级智能体。

PART 04

总结与展望

核心启示

1

范式转变

从“阻塞式工具”思维转向“异步协作伙伴”思维，是构建高级Agent的关键。

2

架构核心

“事件驱动”是实现异步协作的技术基石。

3

落地为王

轻量、务实的工程实践（如声明式工具接入）比追求通用架构更重要。

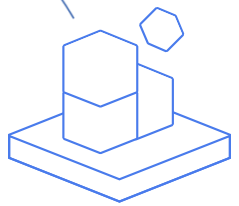
技术创新点总结

分布式协调机制

确保Agent服务高可用且横向可扩展。

事件驱动模型

事件驱动的任务状态推送保证用户及时获知Agent执行进展，增强信任感。



智能工具推荐

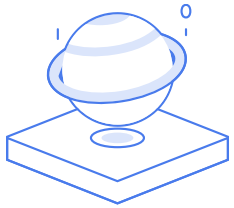
LRU/TOP/AI多维策略融合、上下文感知工具匹配使调用工具更加高效且准确。

分层记忆系统

结构化多层记忆存储实现Agent长短期知识融合。

智能任务编排

从被动执行单步指令升级为主动编排任务的项目经理。



通向更自主的AI协作体

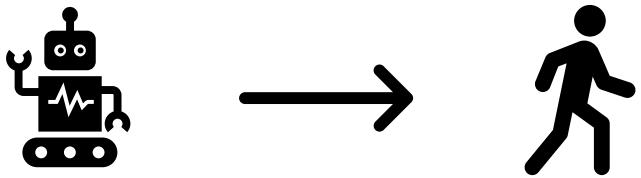
我们的旅程才刚刚开始。

预测性调度

自适应优化

插件自动发现

……



AiDD 8th 2025

第8届AI+研发数字峰会

拥抱AI 重塑研发 AI+ Development Digital Summit

查看会议详情

下一站预告 11/14-15 | 深圳站 12/19-20 | 上海站

深圳站论坛设置

智能装备与机器人

超越“编程 Copilot”

下一代知识工程

智能网联与汽车智能化

AI 测试工具开发与应用

AI 基础设施和运维

数据智能及其行业应用

可信 AI 安全工程

大模型和 AI 应用评测

多 Agent 协同框架

从智能测试到自主测试

大模型推理优化

多模态 LLM 训练与应用

智能化 DevOps 流水线

上下文工程

AiDD

「深行·浅智」

Walk Deep, Think Light.

2025.11.16

AiDD首届麦理浩径徒步

KEYLINK ing

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

AiDD峰会详情

上海站

AI+研发数字峰会

AiDD 峰会

时间：

2025.05.23-24

北京站

AI+研发数字峰会

AiDD 峰会

时间：

2025.08.08-09

深圳站

AI+研发数字峰会

AiDD 峰会

时间：

2025.11.14-15

香港站

「深行·浅智」

AiDD 首届麦理浩径徒步

时间：

2025.11.16

上海站

AI+产品创新峰会

AiDD × 联合峰会

时间：

2025.12.19-20

第7届AI+研发数字峰会
AI+ Development Digital Summit

感谢聆听！

扫码领取会议PPT资料