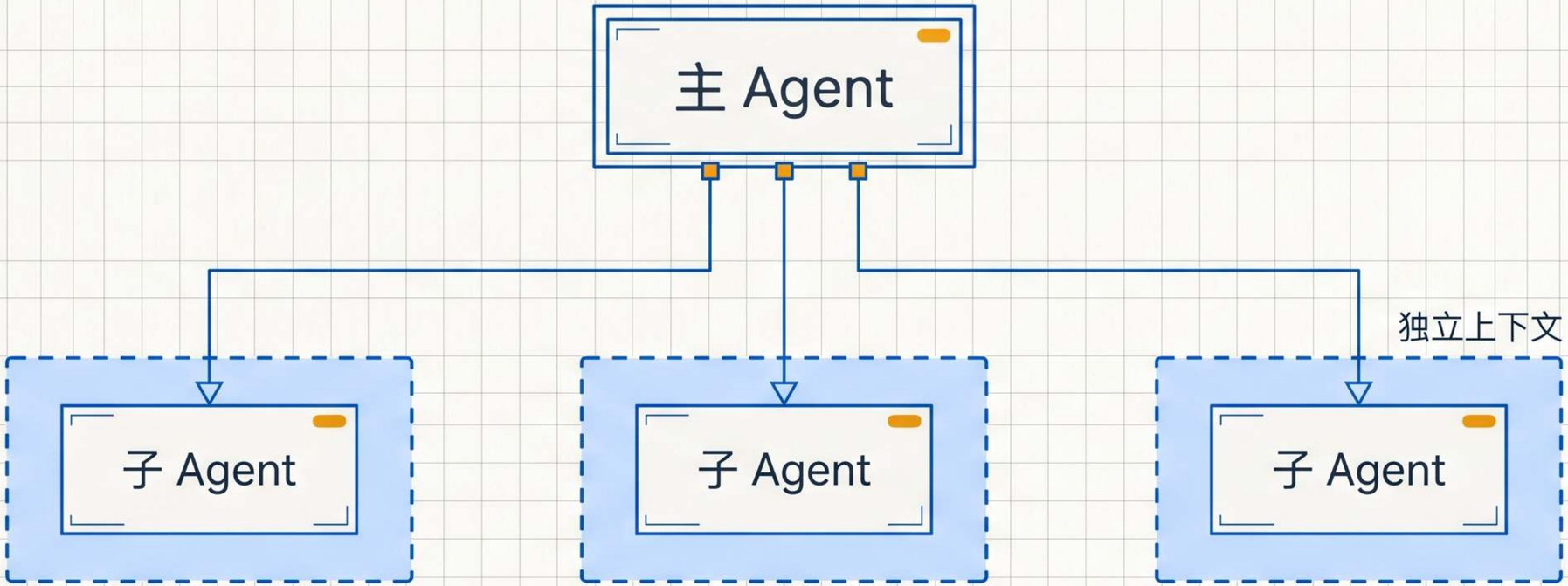
Deep Agents 实战

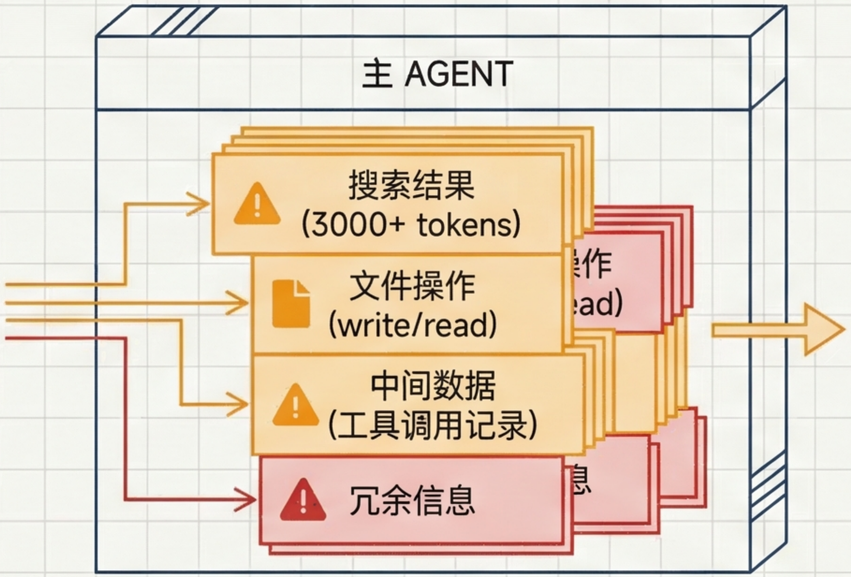
第 7 讲：子 Agent 与上下文隔离



上下文膨胀问题

中间过程正在淹没你的主 Agent

膨胀状态（Bloated State）



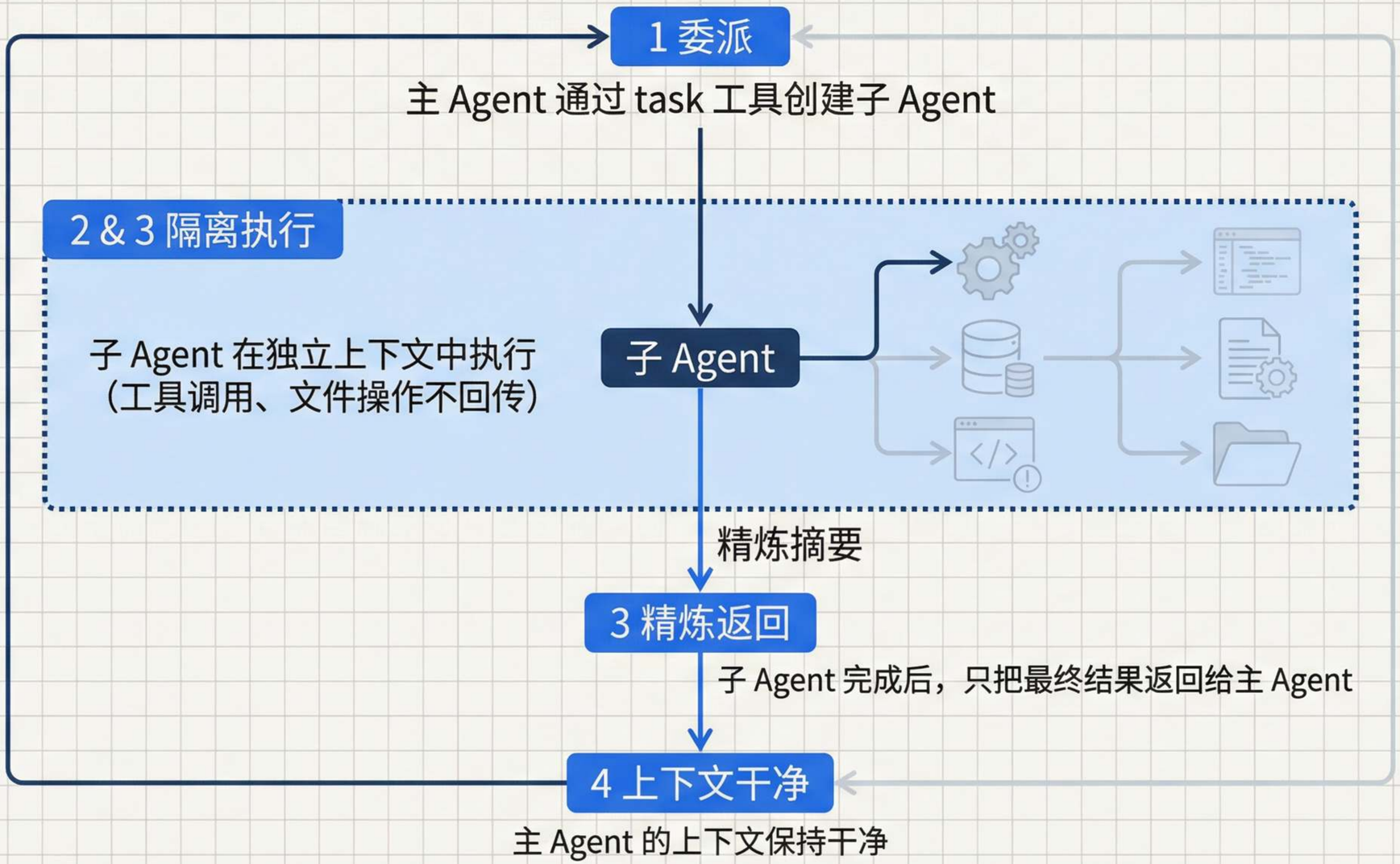
- 一个研究子任务可能涉及：5 次搜索 + 每次 3000+ tokens 结果
- 多次 write_file 保存中间结果、多次 read_file 回顾整理
- 所有工具调用记录堆积在主 Agent 上下文中

理想状态（Ideal State）

- 主 Agent 其实只需要最终的研究摘要

Context Quarantine（上下文隔离）

主 Agent 是项目经理，子 Agent 是专项负责人



什么时候用子 Agent?

决策速查表

场景	推荐
 需要多次搜索和整理的研究任务	✓ 大量中间结果会膨胀上下文
 需要特殊工具或指令的专业任务	✓ 子 Agent 可有自己的工具集
 需要不同模型能力的任务	✓ 子 Agent 可用不同模型
 需要高层协调的复杂任务	✓ 主 Agent 协调，子 Agent 执行
 单步简单查询	✗ 委派开销大于收益
 需要保留中间上下文的任务	✗ 子 Agent 上下文不回传

字典方式定义子 Agent

代码即文档——每个字段一目了然

```python
from deepagents import create_deep_agent

research_subagent = {
    "name": "researcher",                          # ✅ 必填
    "description": "Research assistant",           # ✅ 必填：主 Agent 路由依据
    "system_prompt": "You are a researcher.",      # ✅ 必填，❌ 不继承
    "tools": [web_search],                         # 可选，默认继承 ⚠️ 指定后完全替换
    "skills": ["/skills/research/"],               # 可选，❌ 不继承，独立隔离
}

agent = create_deep_agent(
    model="google_genai:gemini-3.1-pro-preview",
    skills=["/skills/main/"],  # 主 Agent + general-purpose 继承此处
    subagents=[research_subagent],
)
```

system_prompt

❌ 不继承，必须独立定义

tools

✅ 默认继承；显式指定后完全替换（⚠️ 非合并）

skills

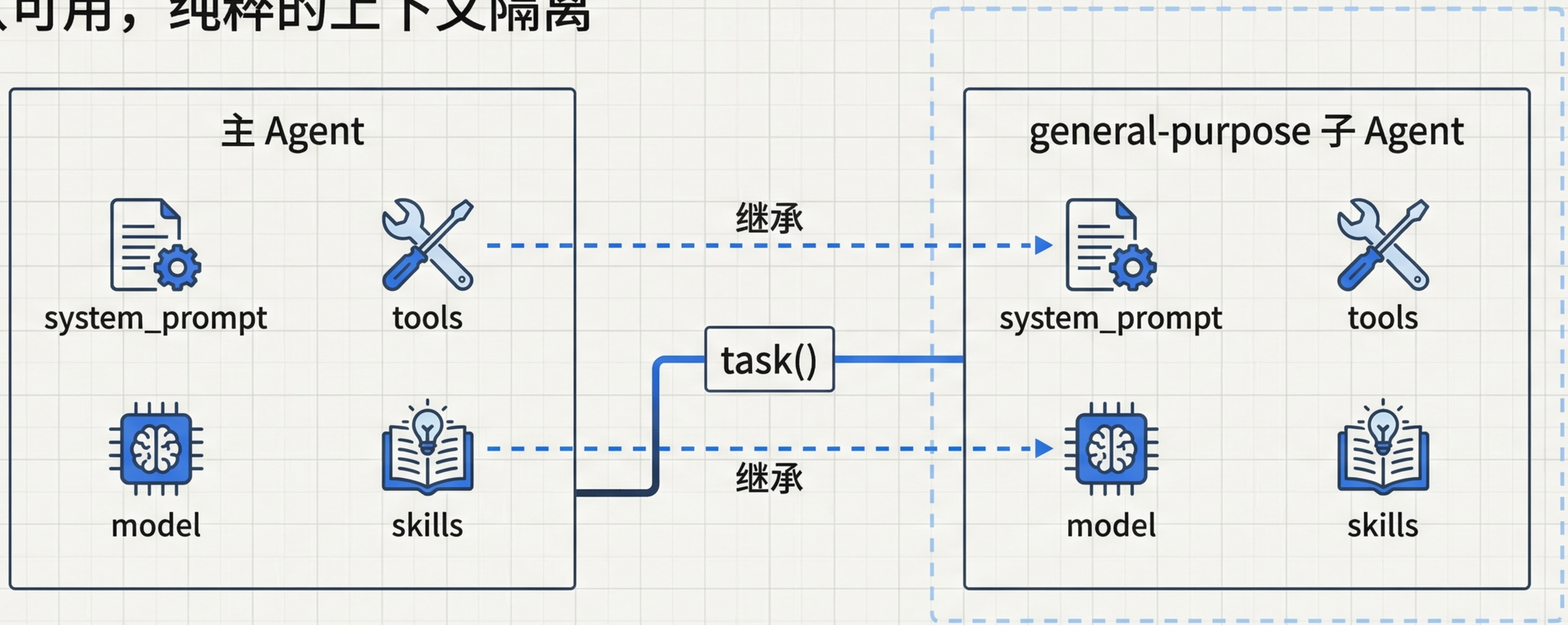
❌ 不继承；仅 general-purpose 子 Agent 自动继承主 Agent skills

model

✅ 支持 'provider:model' 字符串，默认继承

General-purpose 子 Agent

默认可用，纯粹的上下文隔离



唯一的例外：继承主 Agent 的 system_prompt、tools、model、skills

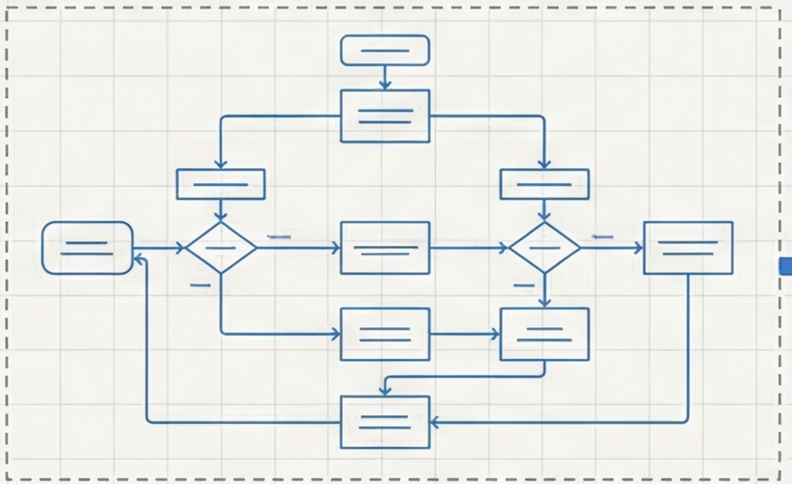
不定义任何子 Agent 也能使用：task(name="general-purpose", task="...")

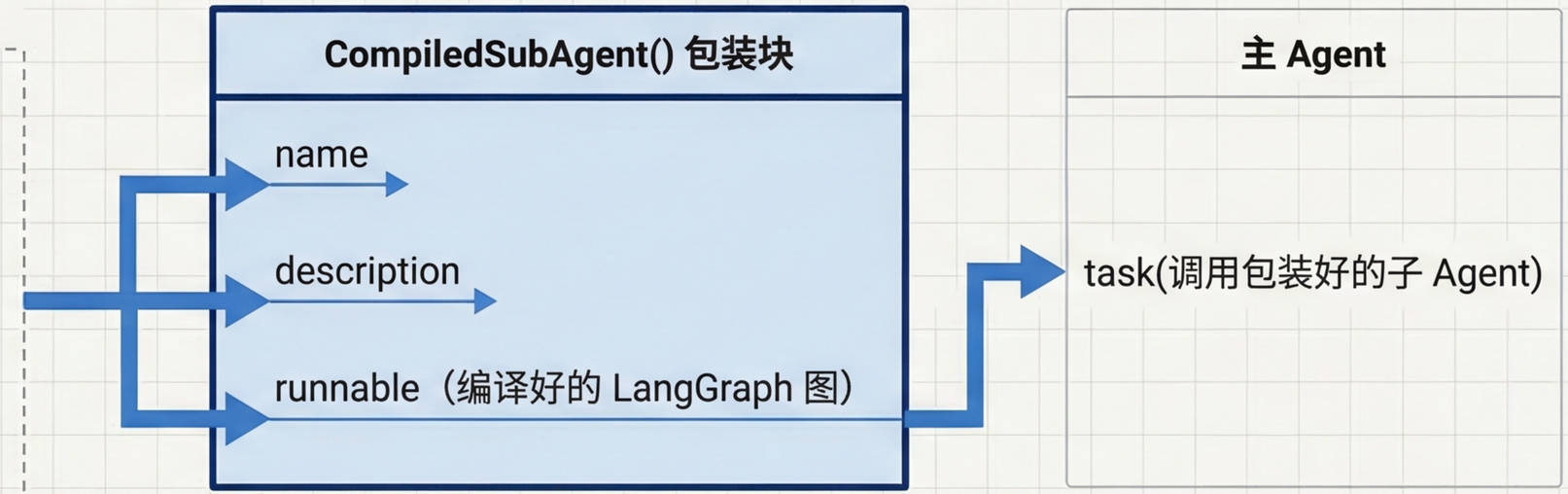
核心价值：能力与主 Agent 相同，但在独立上下文中工作

可覆盖：用 name="general-purpose" 自定义配置

CompiledSubAgent
集成 LangGraph 工作流为子 Agent

预构建 LangGraph 工作流（复杂分支逻辑）



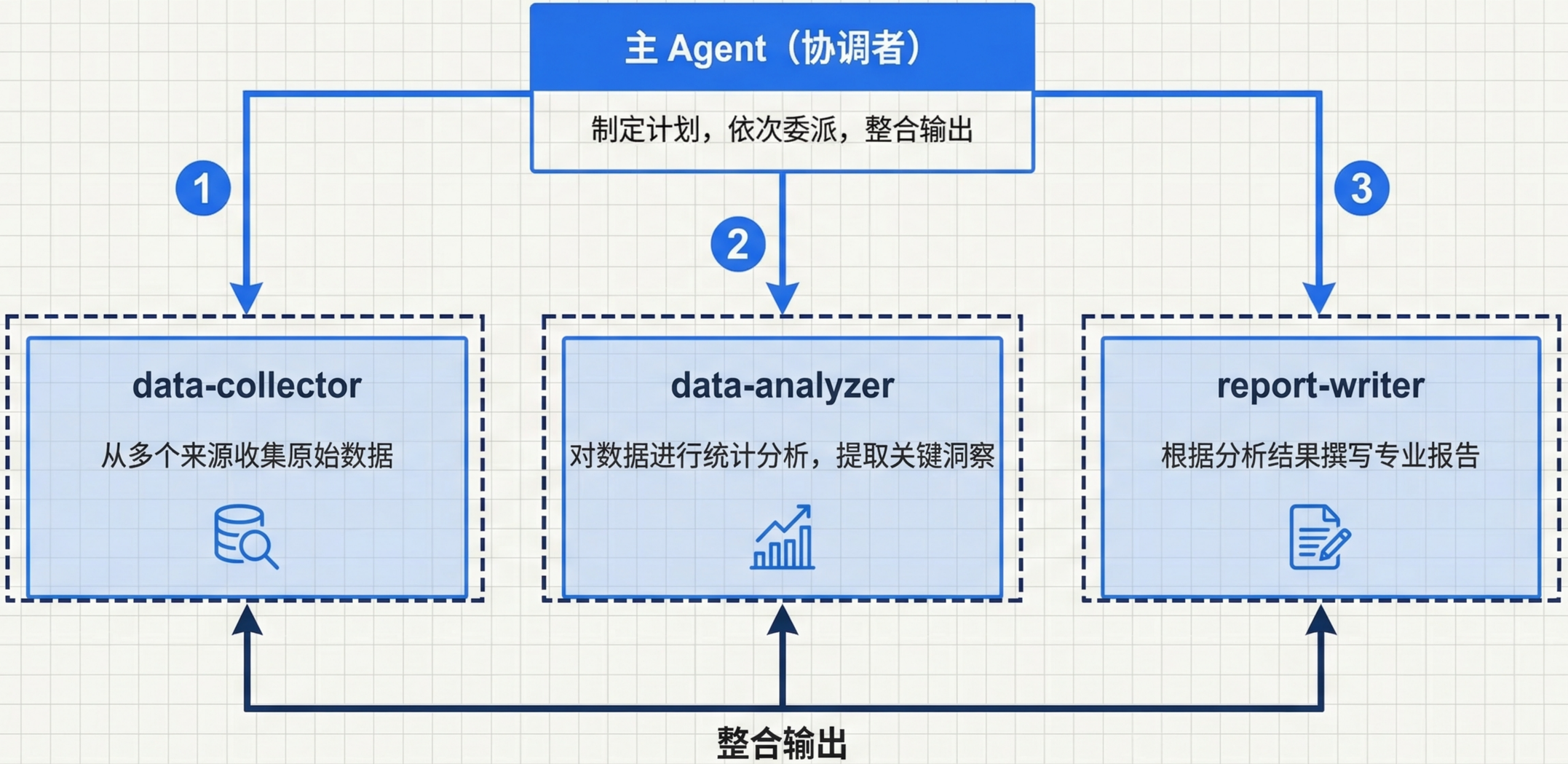


- 适用场景：需要多步骤、有分支逻辑的复杂工作流
- 三个参数：name + description + runnable（编译好的 LangGraph 图）
- 要求：LangGraph 图的 State 必须包含 "messages" 键
- 选择建议：大多数情况用字典方式；有现成 LangGraph 图或需要复杂分支循环时用 CompiledSubAgent

⚠️ 注：State 必须含 "messages" 键

多子 Agent 协作模式

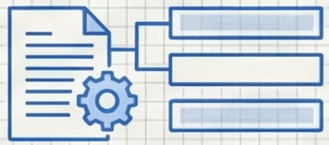
协调者模式：收集 → 分析 → 撰写



子 Agent 最佳实践

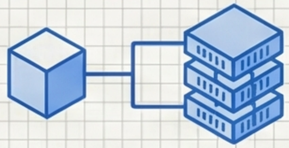
五条关键原则

1 **描述要具体** — 主 Agent 靠 description 决定何时委派及委派给谁

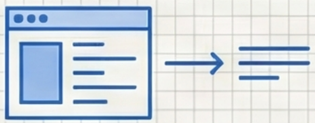


2 **System Prompt 要详细** — 包含输出格式要求和字数限制

3 **工具集要精简** — 最小权限原则，只给必要的工具



4 **不同子 Agent 用不同模型** — 轻量任务用小模型，复杂任务用强模型



5 **返回结果要精练** — 只返回核心内容，否则失去上下文隔离的意义

核心章节回顾 & 进阶预告

第 3 章（第 5-6 讲）

虚拟文件系统

Context Engineering 的核心，6 个工具实现上下文的读写管理

第 4 章（第 7 讲）

任务规划

write_todos 让 Agent 学会拆解和追踪复杂任务

第 5 章（第 7 讲）

子 Agent

Context Quarantine，让主 Agent 学会委派，保持上下文干净

能力闭环（文件系统 + 任务规划 + 子 Agent）

进阶篇

记忆 → 协作 → 技能 → 未来