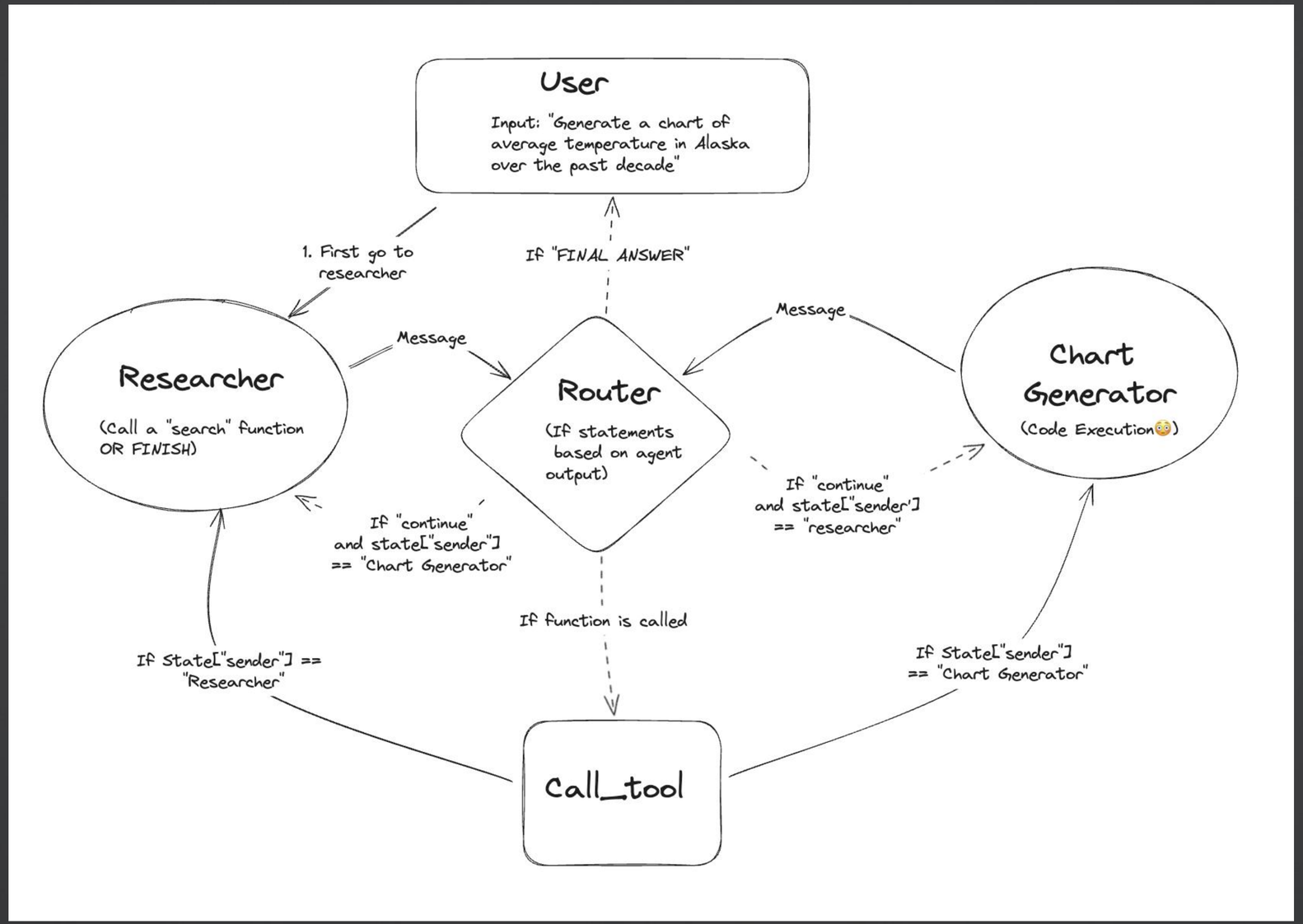
模块十一： 多 Agent 协同自动修复 K8s 故障

王炜/前腾讯云 CODING 高级架构师

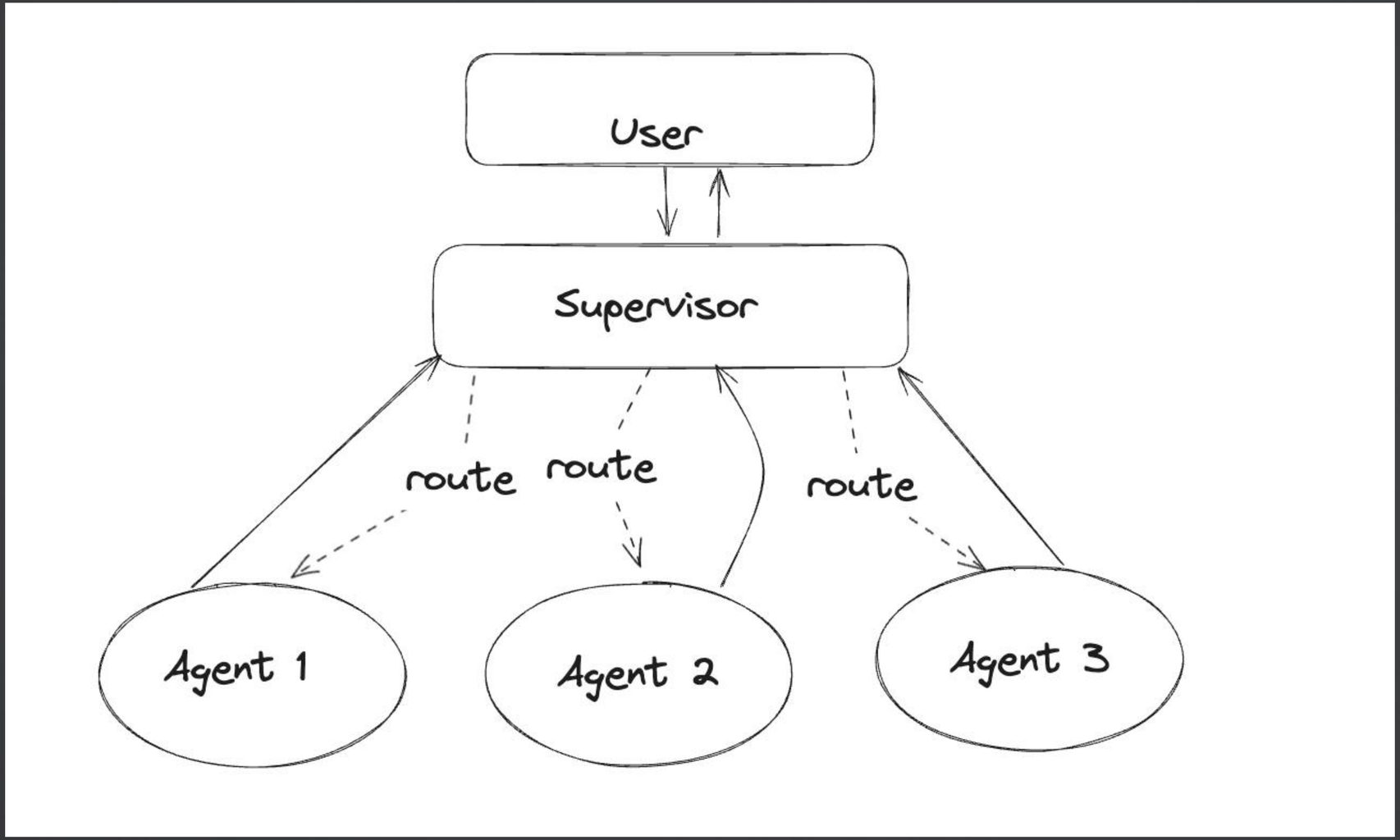
目录

1. 多 Agent 三种协同模式

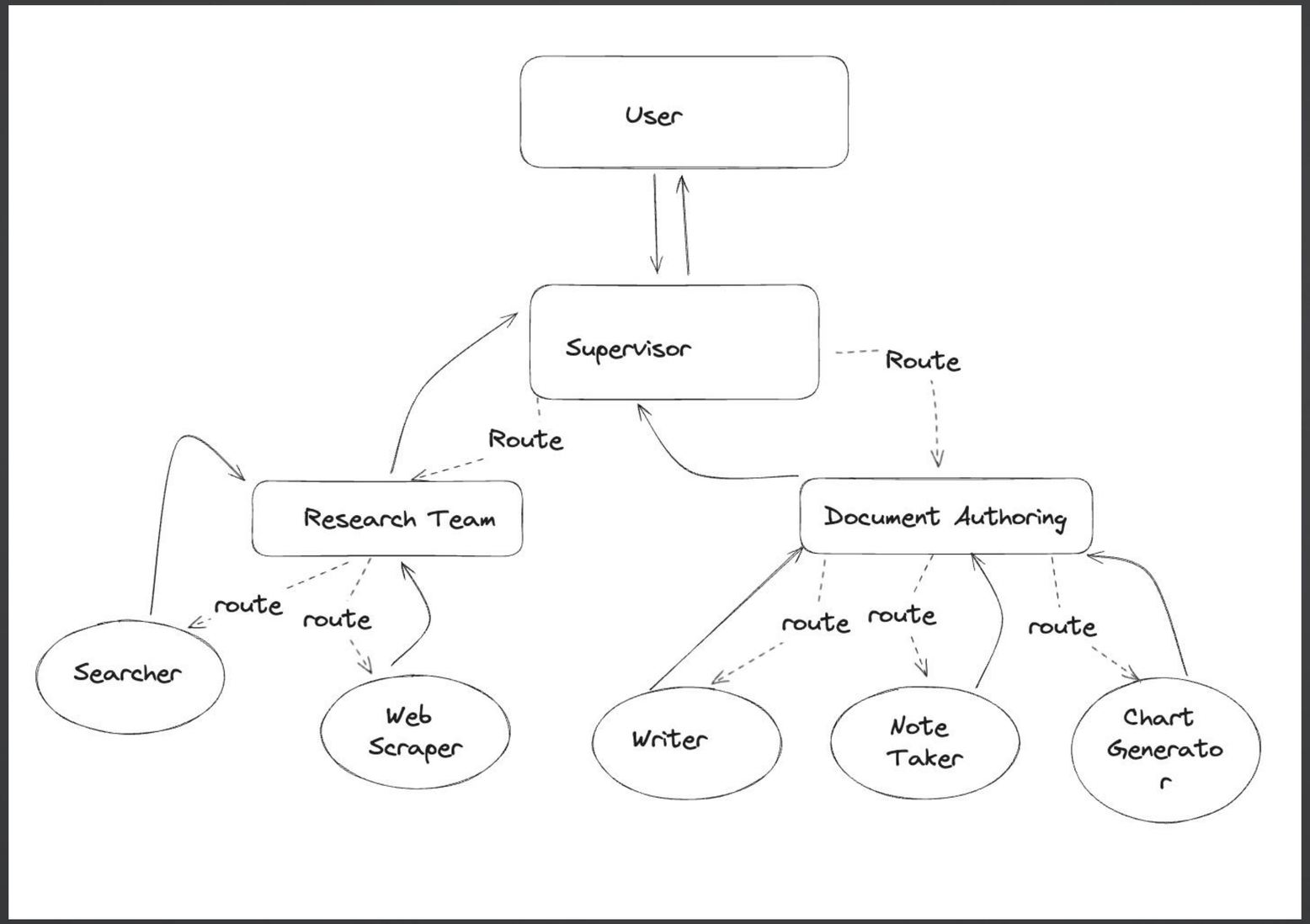
协同模式一：协作


User
Input: "Generate a chart of average temperature in Alaska over the past decade"
1. First go to researcher
If "FINAL ANSWER"
Researcher
(Call a "search" function OR FINISH)
Message
Router
(If statements based on agent output)
Message
Chart Generator
(Code Execution)
If "continue" and state["sender"] == "researcher"
If "continue" and state["sender"] == "Chart Generator"
If function is called
If State["sender"] == "Researcher"
If State["sender"] == "Chart Generator"
Call_tool

协同模式二：管理员模式



协同模式三：分层代理模式

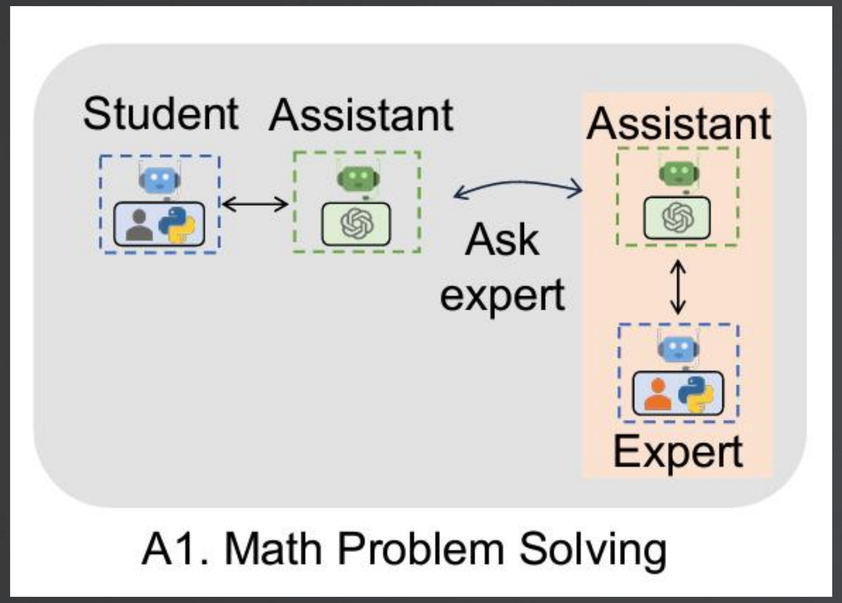


模式对比

对比	协作模式	管理员模式	分层代理模式
实现难度	较简单	一般	复杂
调度器	路由（根据 Agent 的输出）	管理员（主管、LLM）	多层主管
适合任务	Agent 能够自主决策，无需中心化主管调度的简单任务	需要协调多个 Agent 的任务	复杂任务，多层调度

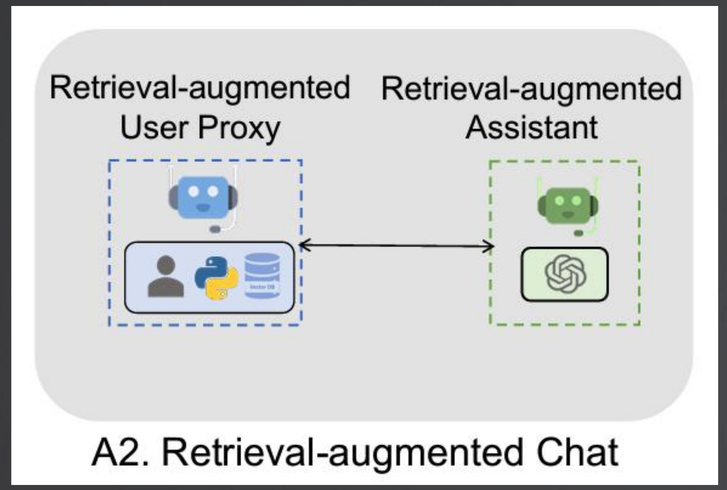
2. 多 Agent 的应用场景

1. 解决数学问题



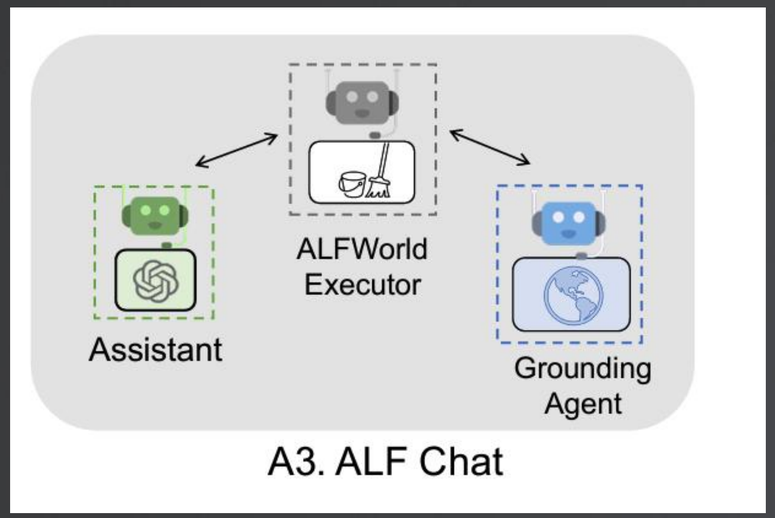
- 由助理和专家 Agent 组成
- 将数学问题交给专门解决该问题的专家，一种数学专家的实现方式是生成用于数学计算的 Python 代码，并交给 Sandbox 运行环境运行生成结果

2. 检索增强聊天



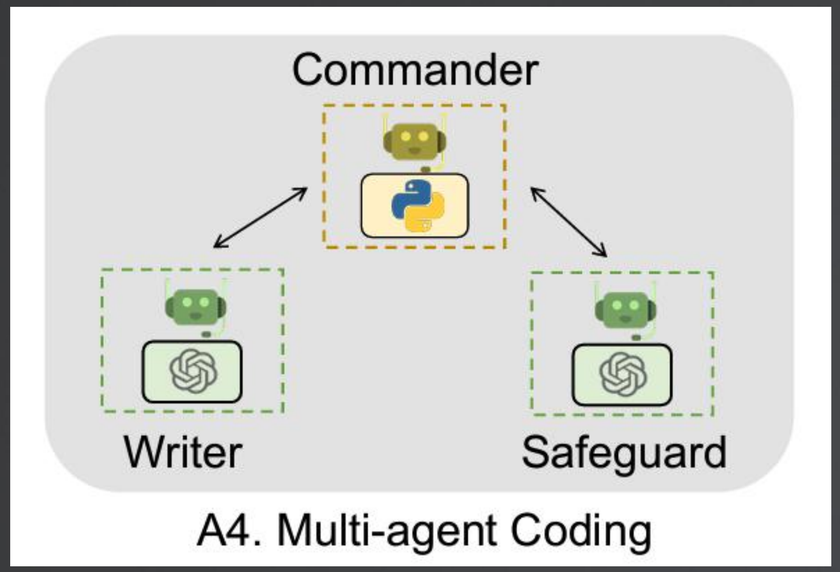
- 由用户代理和检索代理组成
- 当检索到的内容不包含问题的答案时，检索更多的上下文来解决

3. 解决现实世界的问题



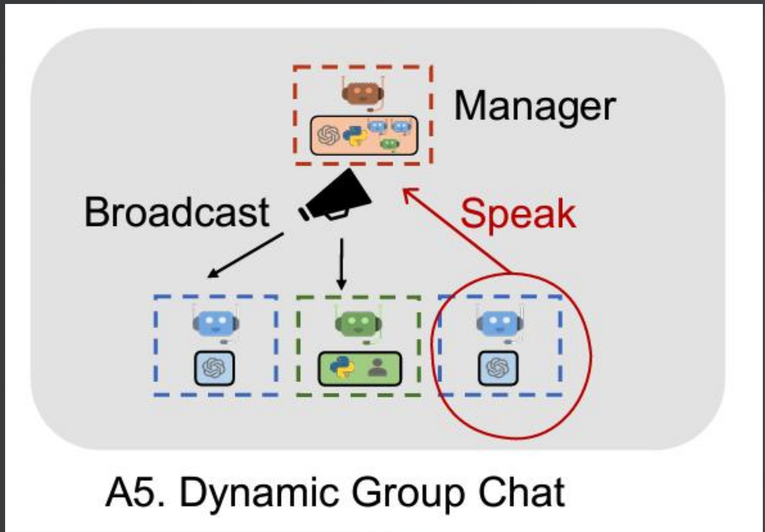
- 由助理代理和执行代理组成
- 助理代理负责规划，执行代理负责执行
- 解决现实世界的常识问题如：「你必须找到并拿走物体，然後才能检查它。你必须先到目标物体的所在位置，然后才能使用它」

4. 多代理编码



- 由指挥官、写代码 Agent 和安全 Agent 组成
- 指挥官将代码 Agent 的结果交给安全 Agent 评估性能和效果，并根据评估结果要求代码 Agent 继续改写

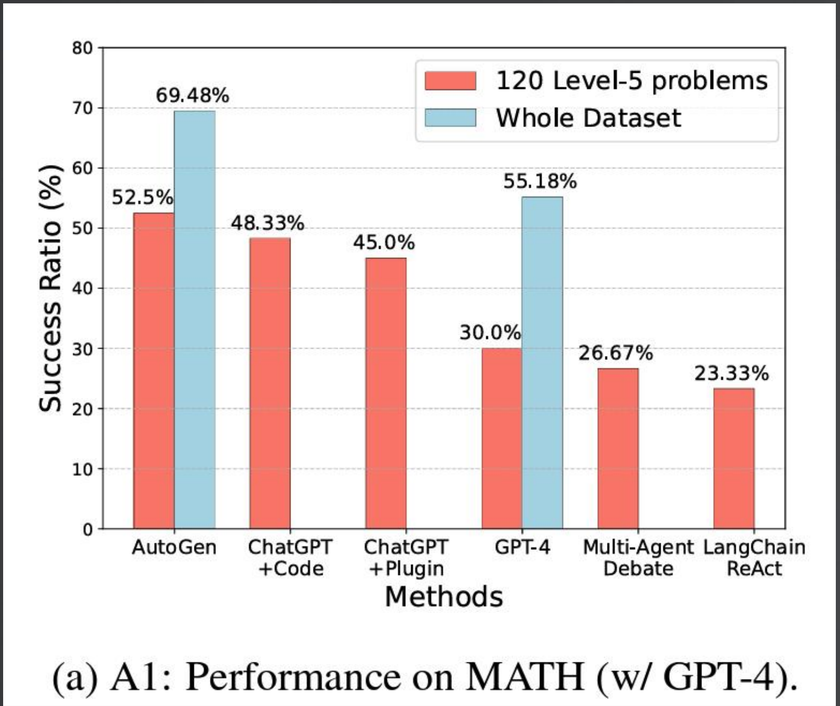
5. 动态组对话



- 由管理员 Agent 和多个发言 Agent 组成
- 管理员选择发言人，然后收集发言人的回复，并将消息广播给其他发言人，让每一个发言人都能了解到全局的上下文

3. 多 Agent 的效果提升

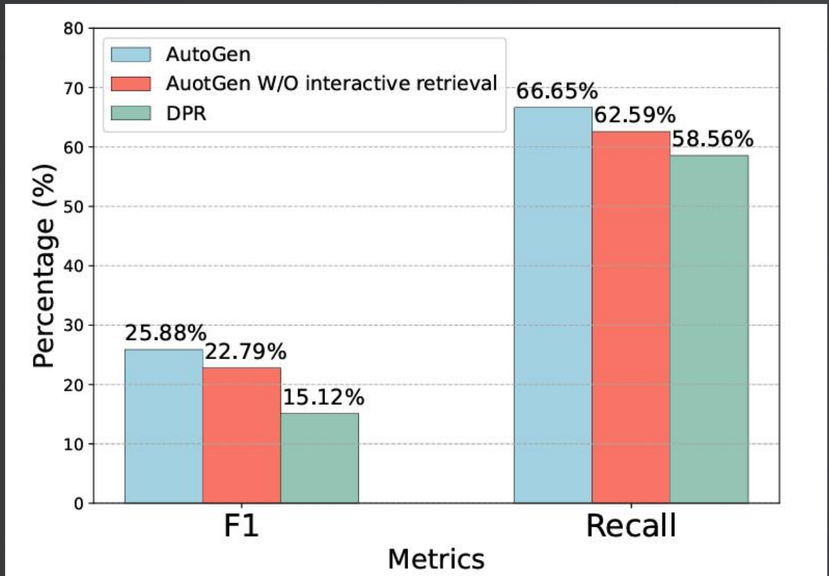
数学问题



(a) A1: Performance on MATH (w/ GPT-4).

- 多 Agent 的方式在解决数学问题上准确度达到 69.48%

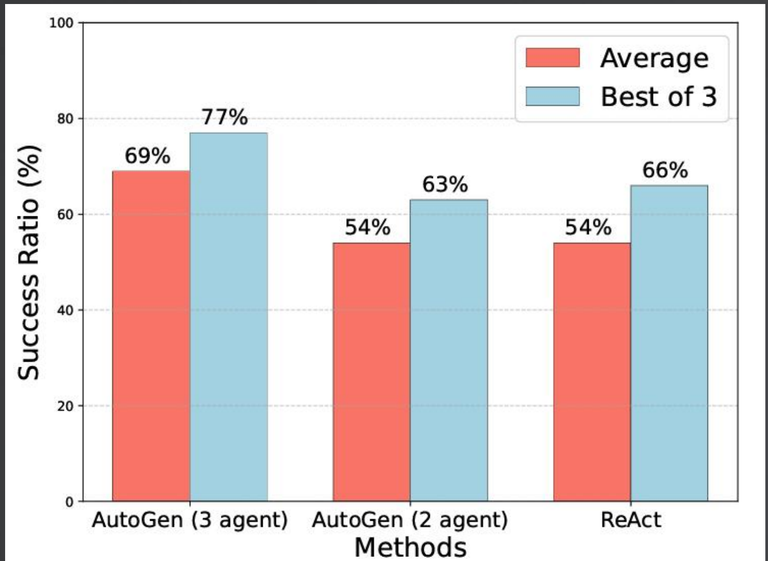
增强检索生成



(b) A2: Q&A tasks (w/ GPT-3.5).

- 多 Agent 的方式在召回准确率达到了 66.65%

增强检索生成

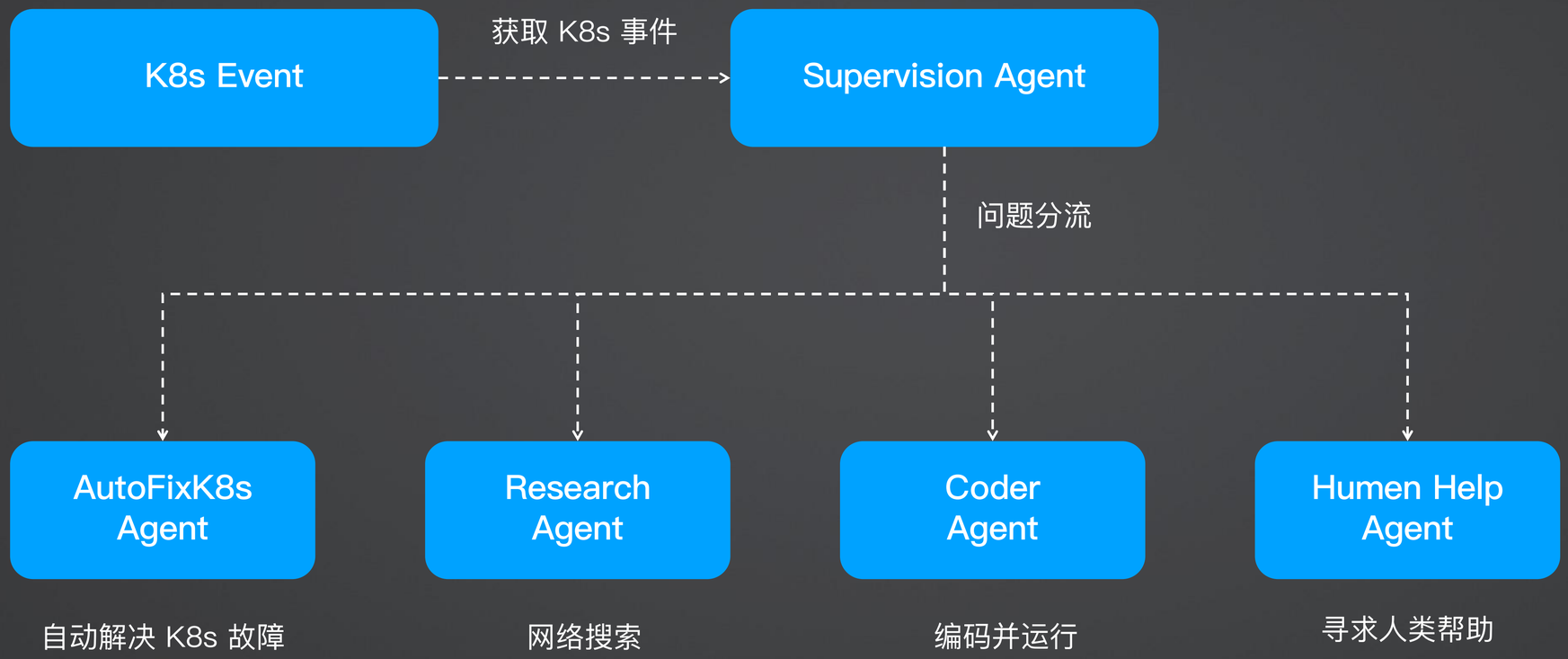


(c) A3: Performance on ALFWorld.

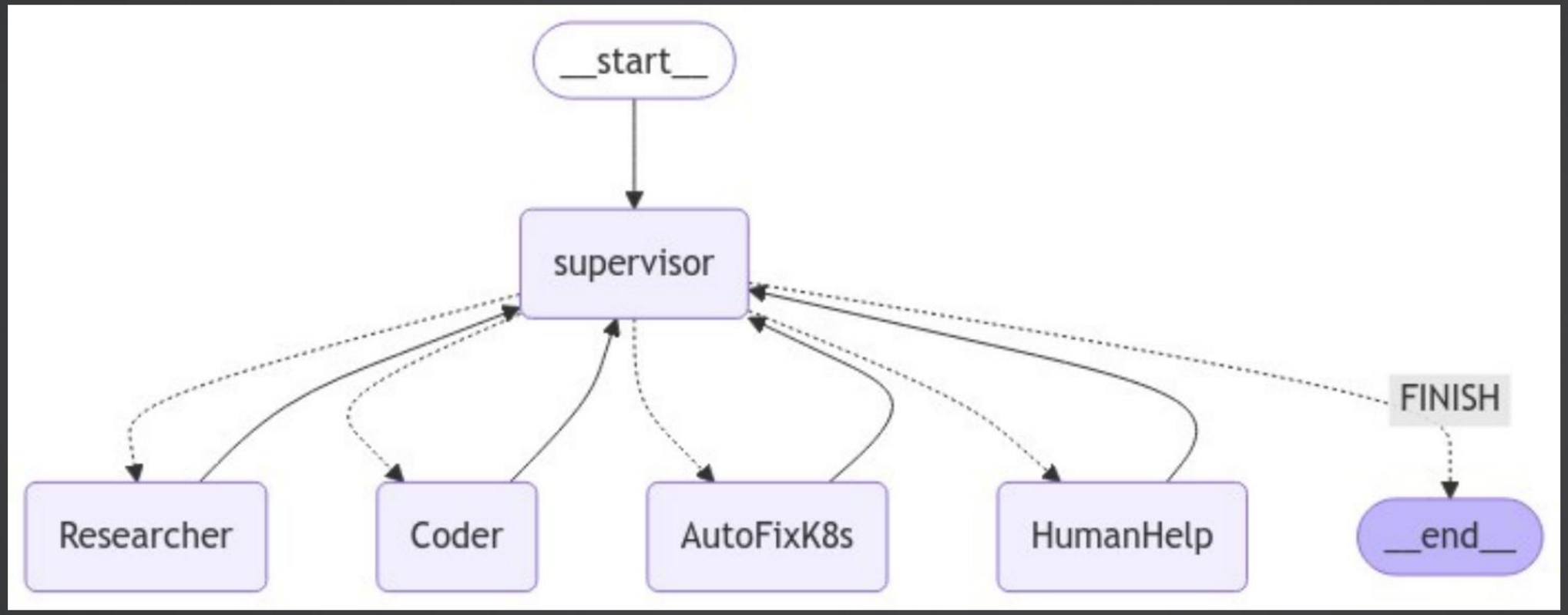
- 在解决现实世界的问题，3 个 Agent 的模式准确率达到了最高

4. 实战：基于多 Agent 协同的 K8s 故障自动修复

架构设计


K8s Event
获取 K8s 事件
Supervision Agent
问题分流
AutoFixK8s Agent
Research Agent
Coder Agent
Humen Help Agent
自动解决 K8s 故障
网络搜索
编码并运行
寻求人类帮助

Graph 事件流



- Supervisor 管理员 Agent 对问题进行分流，并判断问题是否得到解决（FINISH 标识）
- 如果没有，则继续将问题分流给 Agent， 循环直到问题解决
- 如果无法自动解决，则寻求人类帮助（发出通知）

核心提示语：Supervisor 管理员 Agent

untitled.py

```python
# 定义成员以及系统提示词，指示 supervisor 负责调度多个 worker
members = ["Researcher", "Coder", "AutoFixK8s", "HumanHelp"]
system_prompt = (
    "You are a supervisor tasked with managing a conversation between the"
    " following workers: {members}. Given the following user request,"
    " respond with the worker to act next. Each worker will perform a"
    " task and respond with their results and status. When finished,"
    " respond with FINISH."
)
options = ["FINISH"] + members
```

- 设定主管的角色
- 对给定的用户请求进行思考，并将任务分派给不同的 Agent
- 根据 Agent 的响应结果来判断是否需要其他 Agent 协助，或者返回 FINISH 标识标识任务完成

核心提示语：K8s 故障自动解决 Agent

untitled.py

```python
messages=[
    {
        "role": "system",
        "content": "You are a helpful assistant designed to output JSON.",
    },
    {
        "role": "user",
        "content": f"""你现在是一个云原生技术专家，现在你需要根据 K8s 的报错生成一个能通过 kubectl patch 的一段 JSON 内容来修复问题。
K8s 抛出的错误信息是：{event}
工作负载的 YAML 是：
{deployment_yaml}
你生成的 patch JSON 应该可以直接通过 kubectl patch 命令使用，除此之外不要提供其他无用的建议，直接返回 JSON，且不要把 JSON 放在代码块内
""",
    },
],
```

- 基于 Event 事件和工作负载的 YAML 生成 Patch JSON
- 通过代码执行 patch YAML（类似 kubectl patch）达到自动修复故障的目的
- 例如：自动修复镜像版本错误、自动修复 OOM Killed（增加 resource 资源配置）

核心代码：Supervisor Agent

Supervisor.py

```python
# 定义成员以及系统提示词，指示 supervisor 负责调度多个 worker
members = ["Researcher", "Coder", "AutoFixK8s", "HumanHelp"]
system_prompt = (
    "You are a supervisor tasked with managing a conversation between the"
    " following workers: {members}. Given the following user request,"
    " respond with the worker to act next. Each worker will perform a"
    " task and respond with their results and status. When finished,"
    " respond with FINISH."
)
options = ["FINISH"] + members

# 定义 supervisor 的响应类，选择下一个执行的 worker
class routeResponse(BaseModel):
    next: Literal[*options]

# 创建提示模板
prompt = ChatPromptTemplate.from_messages(
    [
        ("system", system_prompt),
        MessagesPlaceholder(variable_name="messages"),
        (
            "system",
            "Given the conversation above, who should act next?"
            " Or should we FINISH? Select one of: {options}",
        ),
    ]
).partial(options=str(options), members=", ".join(members))

# 定义 LLM 模型和 supervisor_agent 函数
llm = ChatOpenAI(model="gpt-4o")

def supervisor_agent(state):
    supervisor_chain = prompt | llm.with_structured_output(routeResponse)
    return supervisor_chain.invoke(state)
```

- 使用 llm.with_structured_output 获取结构化输出，判断下一个需要调度的 Agent
- 使用 supervisor_chain.invoke 方法调用大模型

核心代码：Researcher Agent

Researcher.py

```python
# Researcher Agent
tavily_tool = TavilySearchResults(max_results=5)
research_agent = create_react_agent(llm, tools=[tavily_tool])
research_node = functools.partial(agent_node, agent=research_agent,
name="Researcher")

```

- 使用 Tavily 作为搜索 API 搜索无法解决的问题答案
- 也可以使用 Google API 代替

核心代码：Coder Agent

Coder.py

```python
# 执行 Python 代码的工具，存在安全隐患，需谨慎使用
python_repl_tool = PythonREPLTool()
code_agent = create_react_agent(llm, tools=[python_repl_tool])
code_node = functools.partial(agent_node, agent=code_agent, name="Coder")
```

- 使用 LangChain 内置的 PythonREPLTool 执行 Python 代码
- 注意可能存在安全风险，更好的方式是借助 Docker 充当 Sandbox 环境运行代码

核心代码：AutoFixK8s Agent

AutoFixK8s.py

```python
response = OpenAIClient.chat.completions.create(
  model="gpt-4o",
  response_format={"type": "json_object"},
  messages=[
      {
          "role": "system",
          "content": "You are a helpful assistant designed to output JSON.",
      },
      {
          "role": "user",
          "content": f"""你现在是一个云原生技术专家，现在你需要根据 K8s 的报错生成一个能通过 kubectl patch 的一段 JSON 内容来修复问题。
  K8s 抛出的错误信息是：{event}
  工作负载的 YAML 是：
  {deployment_yaml}
你生成的 patch JSON 应该可以直接通过 kubectl patch 命令使用，除此之外不要提供其他无用的建议，直接返回 JSON，且不要把 JSON 放在代码块内
""",
      },
  ],
)
json_opt = response.choices[0].message.content
# Patch LLM 生成 JSON 到 Deployment
try:
  k8s_apps_v1.patch_namespaced_deployment(
      name=deployment_name, namespace=namespace, body=yaml.safe_load(json_opt)
  )
except Exception as e:
  return "Error when patching deployment" + str(e)
```

- 让大模型生成 Patch JSON 用来修复问题
- 集成 Kubernetes SDK，通过 patch_namespaced_deployment 方法修改工作负载
- 自动修复常见问题

核心代码：Human Help Agent

HumenHelp.py

```python
@tool
def human_help(event_message: str):
    """问题无法解决时寻求人工帮助"""
    url = "https://open.feishu.cn/open-apis/bot/v2/hook/d5e267dc-a92f-43d3-bc45-106b5e718c49"
    headers = {"Content-Type": "application/json"}
    data = {"msg_type": "text", "content": {"text": event_message}}
    response = requests.post(url, headers=headers, data=json.dumps(data))
    return response.status_code
```

- 通过发送飞书消息寻求人类帮助
- 内容包含 K8s 事件

实战编码

- 演示硬编码错误自动修复
  - 镜像版本错误
- 演示寻求人工帮助
- 演示自动获取 K8s Event 自动修复错误
  - 镜像版本错误：编辑 nginx 镜像版本
  - OOM Killed：部署 oom-stress.yaml

实现效果

Supervisor.sh

① Warning Event: ADDED memory-hog-cf8597fbd-st88g - BackOff - Back-off restarting failed container memory-hog in pod memory-hog-cf8597fbd-st88g_default(299da937-f096-4e66-abc2-d9a0e7d62b63)

② {'supervisor': {'next': 'AutoFixK8s'}}

3 ----

③
```
{
  "spec": {
    "template": {
      "spec": {
        "containers": [
          {
            "name": "memory-hog",
            "resources": {
              "limits": {
                "memory": "300Mi"
              }
            }
          }
        ]
      }
    }
  }
}
```

22 {'AutoFixK8s': {'messages': [HumanMessage(content='The issue with the deployment "memory-hog" in the "default" namespace has been successfully fixed. The deployment has been patched to resolve the "BackOff" event.', additional_kwargs={}, response_metadata={}, name='AutoFixK8s')]}}

23 ----

⑤ {'supervisor': {'next': 'FINISH'}}

25 ----

1. 监听到 K8s Warning 事件
2. 思考，将任务调度给合适的 Agent 解决
3. Agent 尝试解决问题，生成 Patch JSON
4. Agent 成功解决问题，将消息返回给 Supervisor
5. Supervisor 收到消息，返回 FINISH 标识，退出循环

课后作业

- 尝试提出两个问题，实现将任务调度给 Coder Agent 和 Researcher Agent
- 解耦 AutoFixK8s Agent，创建一个新的 Agent，命名为 GetK8sYAML Agent，将获取 YAML 的功能拆分至该 Agent，并将串联自动修复流程，截图

THANKS