

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



Tool Use for Language Model Agents

How models turn token predictions into actions

Carnegie Mellon University
Language Technologies Institute

Graham Neubig
Language Technologies Institute

1

从词元预测到行动

Graham Neubig

语言技术研究所

语言模型智能体的工具使用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



基本思路

基本思路

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Tool definition

A tool is an interface through which a language model can invoke an external computer program.

- Search
- Calculator
- Browser
- Python
- API

The model proposes; external software decides whether and how to execute.

What Are Tools Anyway? · Wang et al., COLM 2024 ↗

工具是语言模型调用外部计算机程序的一种接口。

工具示例：搜索、计算器、浏览器、Python、API。

模型提出请求；外部软件决定是否执行以及如何执行。

《工具究竟是什么？》· Wang 等, COLM 2024 ↗

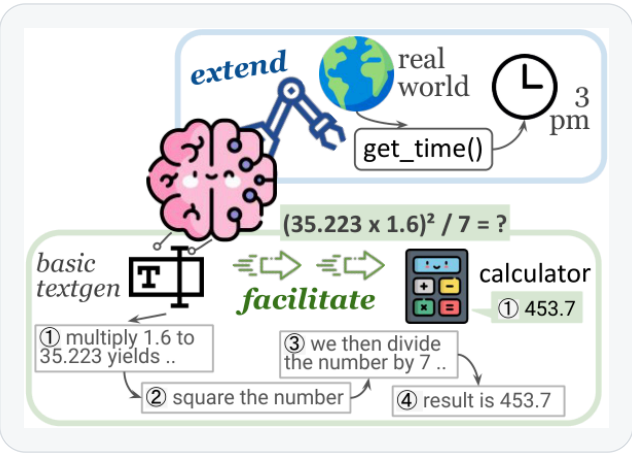
工具定义

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Benefits of tools



Extend

Access information or actions unavailable in the model's parameters.

Facilitate

Delegate operations external programs perform more reliably or efficiently.

扩展：获取模型参数中没有的信息或行动能力。

促进：把操作委托给外部程序，使其更可靠或更高效地完成。

《工具究竟是什么？》· Wang 等, COLM 2024 ↗

What Are Tools Anyway? · Wang et al., COLM 2024 ↗

4

工具的好处

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Tool-use scenarios

Need	Example tool	Benefit
Current information	Search	Fresh evidence
Exact computation	Calculator or Python	Reliable execution
Private state	Account API	Authorized application data
External action	Browser or API	Changes the environment

Call tools only when their benefit outweighs latency, cost, failure, and risk.

5

需求 | 示例工具 | 收益

当前信息 | 搜索 | 新鲜证据。

精确计算 | 计算器或 Python | 可靠执行。

私有状态 | 账户 API | 经授权的应用数据。

外部行动 | 浏览器或 API | 改变环境。

只有当工具收益高于延迟、成本、失败和风险时才调用工具。

工具使用场景

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



工具示例

工具示例

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Building ChatGPT

What sorts of things can you help me do?

I can answer questions, find information, run code, create images, and complete specific workflows.

What calls are necessary?

Start from the abilities we want, then define a narrow interface for each kind of interaction.

7

‘你能帮我做什么事？’ ——我可以回答问题、查找信息、运行代码、创建图像，并完成特定工作流。

‘需要哪些调用？’ ——先从期望的能力出发，再为每种交互定义范围明确的接口。

构建 ChatGPT

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Textual responses

```
finish(text)
```

Return the final text and stop the agent loop. This is a harness control action, not an external program.

What is the capital of Ghana?

CONTROL finish("Accra is the capital of Ghana.")
CALL

Accra is the capital of Ghana.

8

‘加纳的首都是哪里？’

‘finish(text)’：返回最终文本并停止智能体循环。

这是执行框架的控制操作，不是外部程序。

调用：‘finish("Accra is the capital of Ghana.")’。

回复：阿克拉是加纳的首都。

文本回复

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Information retrieval

search_web(query)

Retrieve information from the web.

What did CMU announce today?

TOOL CALL

search_web("site:cmu.edu announced today")

TOOL RESULT

Three matching news items...

CMU announced three items today...

9

信息检索

‘CMU 今天宣布了什么？’

工具调用：`search\_web("site:cmu.edu announced today")`。

`search\_web(query)`：从网络检索信息。

工具结果：找到三条匹配的新闻.....

回复：CMU 今天宣布了三项内容.....

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Code execution

```
execute_code(code
)
```

Run a program in an isolated environment and return its output.

What is the mean of 18, 23, 41, and 54?

TOOL CALL

execute_code("sum([18,23,41,54])/4")

TOOL RESULT

34.0

The mean is 34.

10

代码执行

‘18、23、41 和 54 的平均值是多少？’

‘execute_code(code)’：在隔离环境中运行程序并返回其输出。

工具调用：‘execute_code("sum([18,23,41,54])/4")’；工具结果：‘34.0’。

回复：平均值为 34。

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Image generation

`generate_image(query)`

Generate an image from a textual description.

Create a watercolor robot studying in a library.

TOOL CALL

`generate_image("watercolor robot studying in a library")`

TOOL RESULT

`generated_image.png`

Here is the generated image.

11

图像生成

‘创作一幅水彩画：机器人正在图书馆学习。’

`generate\_image(query)`：根据文本描述生成图像。

工具调用：`generate\_image("watercolor robot studying in a library")`；工具结果：`generated\_image.png`。

回复：这是生成的图像。

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Custom functions

`create_grocery_cart(items)`

Expose an application-specific capability through one typed call.

Make a cart with bananas, milk, and coffee.

TOOL CALL

`create_grocery_cart(["bananas","milk","coffee"])`

TOOL RESULT

`Cart C17 · 3 items · $31.42`

I created a three-item cart totaling \$31.42.

自定义函数

‘把香蕉、牛奶和咖啡加入购物车。’

‘create_grocery_cart(items)’：通过一次带类型的调用暴露特定于应用的能力。

工具调用：‘create_grocery_cart(["bananas","milk","coffee"])’；工具结果：购物车 C17 · 3 件商品 · \$31.42。

回复：我创建了一个包含 3 件商品、总额为 \$31.42 的购物车。

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

以编程方式调用工具

Programmatic tool calling

13

以编程方式调用工具

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Code as a meta-tool

build-in functions → tools

Q: The bakers baked 200 loaves of bread ... How many loaves of bread did they have left?

LM The bakers started with 200 loaves. They sold 93 in the morning ... The answer is 62.

loaves_baked, loaves_returned = 200, 6
sold_morning, sold_afternoon = 93, 39
answer = loaves_baked - loaves_sold_morning - loaves_sold_afternoon + loaves_returned

CodeLM

external libraries → tools

Q: The table shows how many... What is the max number of vacation days across years?

Year	2013	2014	2015
Vacation days	23	18	11

import pandas as pd
df = pd.DataFrame({"Year": [2013, 2014, 2015], "Vacation days": [23, 18, 11]})
max_days = df["Vacation days"].max()

CodeLM

utility functions → tools

Q: Who is wearing the dress?



dress_box = locate_objects(image, "dress")
dress_region = crop_region(image, dress_box)
answer = visual_qa(dress_region, question="Who is wearing the dress?")

CodeLM

Code composes control flow, libraries, and multiple operations in one action.

内置函数 → 工具；外部库 → 工具；实用函数 → 工具。

问题：面包师烤了 200 条面包.....还剩多少片？语言模型回答：面包师早上开始时有 200 条面包.....卖出 93 条.....答案是 62。CodeLM 的代码将 `loaves\_baked = 200`、`loaves\_returned = 6`、`sold\_morning = 93`、`sold\_afternoon = 39`，并计算 `loaves\_baked - loaves\_sold\_morning - loaves\_sold\_afternoon + loaves\_returned`。

问题：表格显示多年数据.....各年中休假天数的最大值是多少？表中年份为 2013、2014、2015；休假天数为 23、18、11。CodeLM 导入 pandas，构造 DataFrame，并用 `df["Vacation days"].max()` 求最大值。

问题：谁穿着这条裙子？CodeLM 用 `locate\_objects(image, "dress")` 定位衣物，再用 `crop\_region(image, dress\_box)` 裁剪区域，最后调用 `visual\_qa(dress\_region, question=...)` 回答。

代码可以在一次行动中组合控制流、库和多个操作。

《工具究竟是什么？》· Wang 等, COLM 2024 ↗

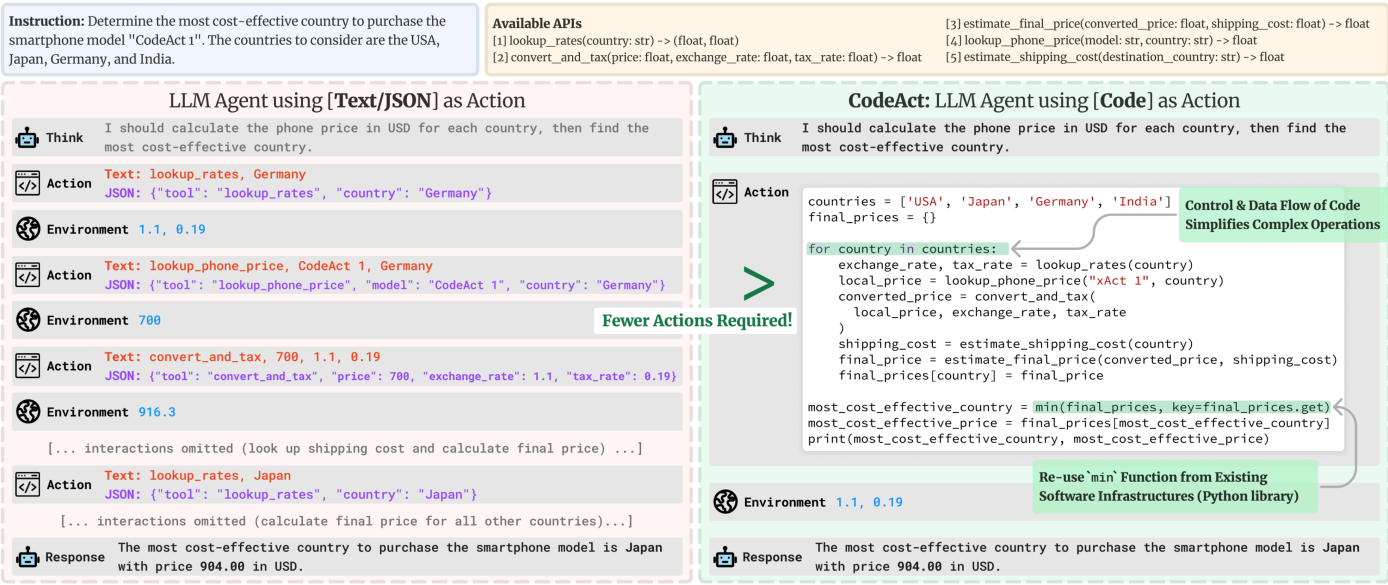
代码作为元工具

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Programmatic composition



CodeAct · Wang et al., ICML 2024 ↗

指令：确定购买智能手机型号 ‘CodeAct 1’ 性价比最高的国家。考虑国家为美国、日本、德国和印度。

可用 API：`lookup\_rates(country)` → (float, float); `convert\_and\_tax(price, float, exchange\_rate: float, tax\_rate: float)` → float; `estimate\_final\_price(converted\_price: float, shipping\_cost: float)` → float; `lookup\_phone\_price(model: str, country: str)` → float; `estimate\_shipping\_cost(destination: str)` → float。

文本 / JSON 作为行动的 LLM 智能体：先调用 `lookup\_rates` 查询德国汇率，再调用 `lookup\_phone\_price` 查询 CodeAct 1 在德国的价格，调用 `convert\_and\_tax`，之后还需查询运费并计算最终价格；中间省略了多次交互及其他国家的计算。最后回答：购买 CodeAct 1 性价比最高的国家是日本，价格为 904.00 美元。

CodeAct：使用代码作为行动的 LLM 智能体。代码枚举 `['USA', 'Japan', 'Germany', 'India']`，循环调用汇率、手机价格、税费和运费函数，保存各国最终价格，并用 `min(final\_prices, key=final\_prices.get)` 选出最低价。结果同样是日本，价格为 904.00 美元。

代码的控制流和数据流简化了复杂操作；可以复用现有软件基础设施（Python 库）中的 `min` 函数。

CodeAct · Wang 等, ICML 2024 ↗

译注 / 与原文翻译分开

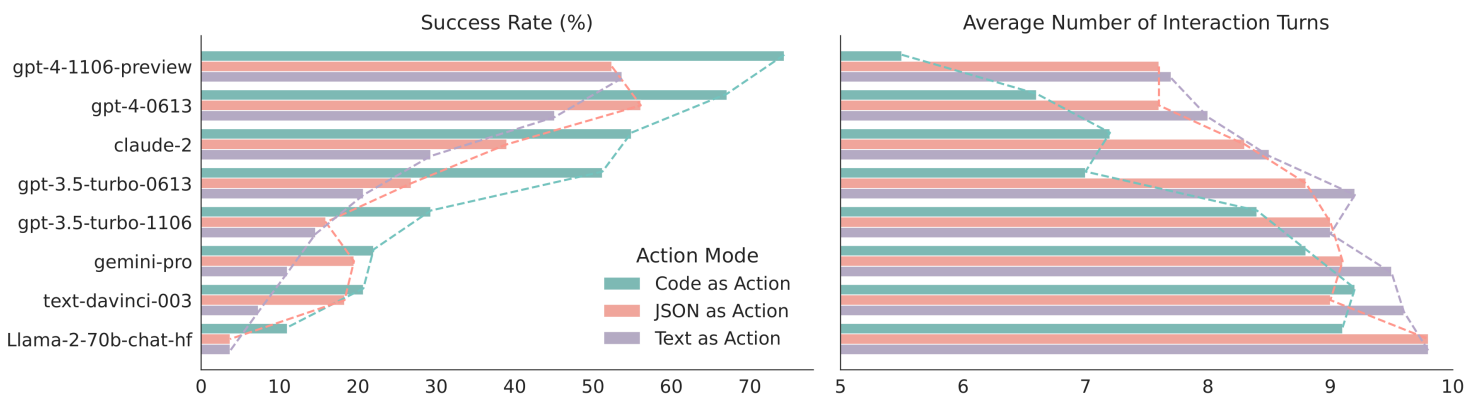
图中有省略的多步交互，未显示省略部分的具体工具调用内容。

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

CodeAct results



In the displayed subset, code had the best success rate for 8/8 models.

Code used the fewest turns for 7/8; across all 17 models, both counts are 12/17.

CodeAct · Wang et al., ICML 2024 ↗

16

CodeAct 的结果。

图表：成功率（%）；平均交互轮数。模型：`gpt-4-1106-preview`、`gpt-4-0613`、`claude-2`、`gpt-3.5-turbo-0613`、`gpt-3.5-turbo-1106`、`gemini-pro`、`text-davinci-003`、`Llama-2-70b-chat-hf`。行动模式：代码作为行动、JSON 作为行动、文本作为行动。

所展示的子集中，代码在 8/8 个模型上取得最高成功率。

代码在 8 个模型中的 7 个上使用的轮数最少；在全部 17 个模型中，这两项计数均为 12/17。

CodeAct · Wang 等, ICML 2024 ↗

译注 / 与原文翻译分开

图表横轴刻度可读，但柱形未标注精确数值；不根据图形估读各模型的具体成功率或轮数。

CodeAct 的结果

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Code is a high-power tool

Expressive

Loops · variables · libraries

Good for multi-step data manipulation and composition.

Harder to constrain

Broad action space

Validation is less precise than for a narrow typed function.

Higher impact

Files · network · processes

Use sandboxing, resource limits, permissions, and audit logs.

17

表达能力强：循环、变量、库；适合多步数据处理与组合。

更难约束：行动空间广；与范围明确的类型化函数相比，验证不够精确。

影响更大：可访问文件、网络 and 进程；应使用沙箱、资源限制、权限和审计日志。

代码是一种高能力工具

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

提供工具的机制



提供工具的机制

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Tool calls as tokens

Text continuation

The weather is sunny.

Tool continuation

```
<tool_call>
get_weather( ... )
```

The model requests an action. The harness executes it.

19

文本续写： ‘天气晴朗。’

工具续写：`<tool\_call> get\_weather(...)`。

模型请求执行一项行动；执行框架执行该行动。

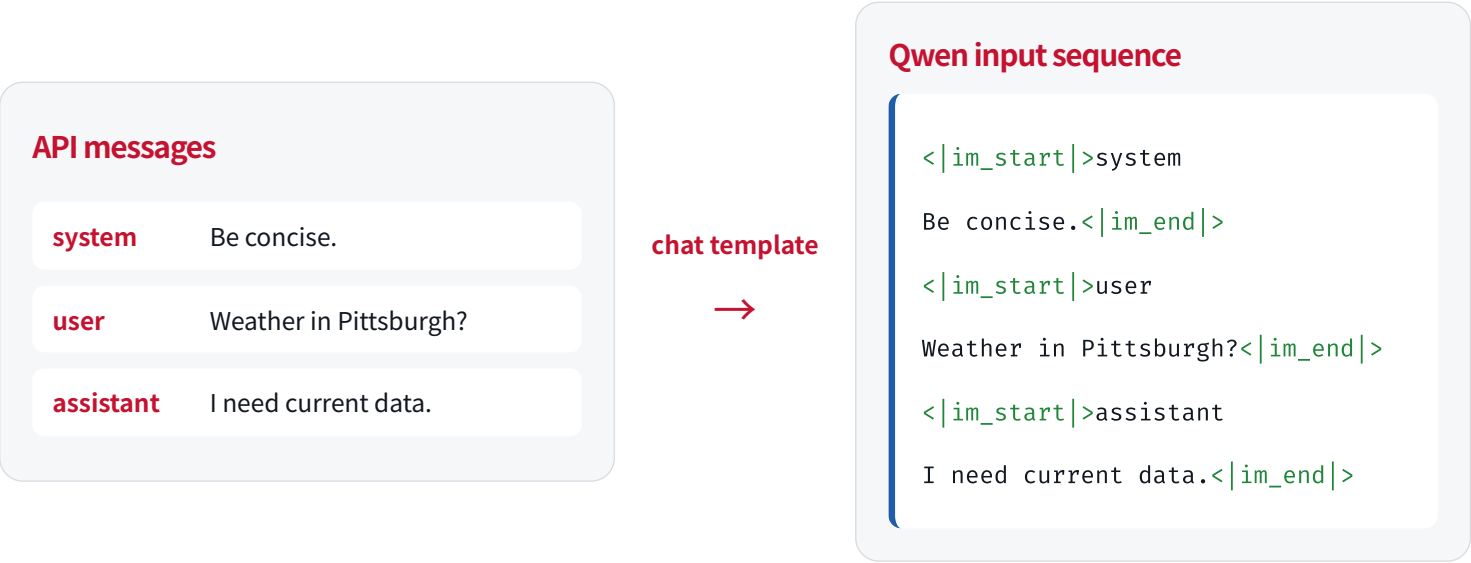
作为词元的工具调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Quick review: messages become model input



Roles structure the API; the template serializes one model-input sequence.

20

API 消息：system： ‘简洁作答。’ ； user： ‘匹兹堡的天气怎么样？’ ； assistant： ‘我需要当前数据。’

Qwen 输入序列经聊天模板序列化为：`<|im\_start|>system` 简洁作答。`<|im\_end|>`；`<|im\_start|>user` 匹兹堡的天气怎么样？`<|im\_end|>`；`<|im\_start|>assistant` 我需要当前数据。`<|im\_end|>`。

角色组织 API 消息；模板将其序列化为一段模型输入序列。

快速回顾：消息成为模型输入

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Tools add structure to the request

Tool definition

```
{
  "type": "function",
  "function": {
    "name": "get_weather",
    "parameters": {
      "type": "object",
      "properties": {"city": {"type": "string"}},
      "required": ["city"]
    }
  }
}
```

API request

```
response =
client.chat.completions.create(
  model=model,
  messages=messages,
  tools=[weather_tool],
)
```

The harness sends messages and tool schemas; the template exposes both to the model.

21

工具定义：类型为 `function`，函数名 `get\_weather`；参数为对象，属性 `city` 的类型为字符串，且 `city` 为必需字段。

API 请求：`client.chat.completions.create(model=model, messages=messages, tools=[weather\_tool])`。

执行框架向模型发送消息和工具 Schema；模板将二者提供给模型。

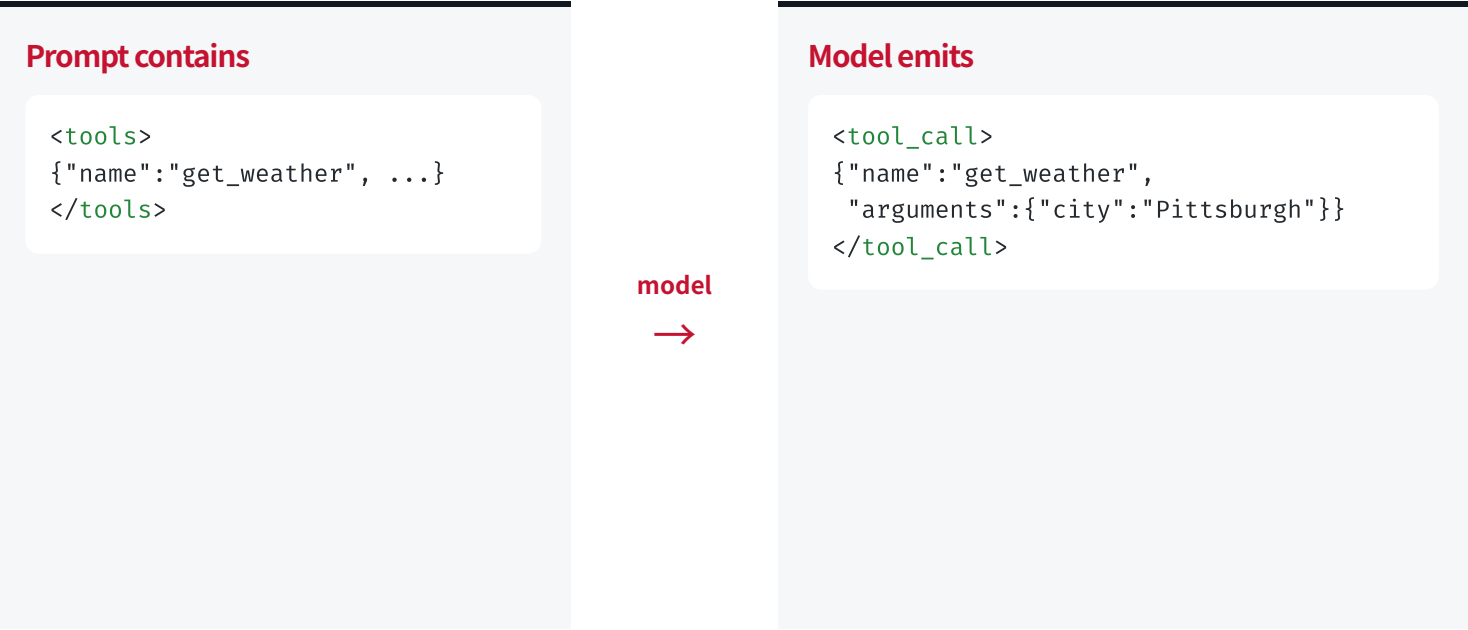
工具为请求增加结构

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Definition in, tool call out



The model emits the call content; the API response attaches a unique call ID.

22

提示中包含：`<tools>{"name": "get\_weather", ...}</tools>`。

模型生成：`<tool\_call>{"name": "get\_weather", "arguments": {"city": "Pittsburgh h"}}</tool\_call>`。

模型生成调用内容；API 响应会附上唯一的调用 ID。

输入定义，输出工具调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Same tool call, different model protocols

Qwen 3.8

Function + parameter tags

```
<|im_start|>assistant
<tool_call>
<function=get_weather>
<parameter=city>
Pittsburgh
</parameter>
</function>
</tool_call><|im_end|>
```

Mistral Small 4

Control token + call ID

```
[TOOL_CALLS] [{"name":
"get_weather",
"arguments": {"city":
"Pittsburgh"}, "id":
"abc123xyz"}]</s>
```

DeepSeek V3.2

DSML invocation block

```
< | DSML | function_calls>
< | DSML | invoke
name="get_weather">
< | DSML | parameter
name="city"
string="true">Pittsburgh<
/ | DSML | parameter>
</ | DSML | invoke>
</ | DSML | function_calls>
```

One schema; model-specific serialization and parsing.

23

Qwen 3.8：函数与参数标签。Mistral Small 4：控制词元与调用 ID。DeepSeek V3.2：DSML 调用块。

示例均调用 `get\_weather`，参数为 `city=Pittsburgh`。Qwen 使用 `<|im\_start|>assistant`、`<tool\_call>`、`<function=get\_weather>` 和 `<parameter=city>` 等标签；Mistral 使用 `[TOOL\_CALLS]` 并附 `id: "abc123xyz"`；DeepSeek 使用 `<DSML function\_calls>`、`<DSML invoke name="get\_weather">` 和 `<DSML parameter name="city" string="true">` 等标记。

同一 Schema；序列化和解析方式因模型而异。

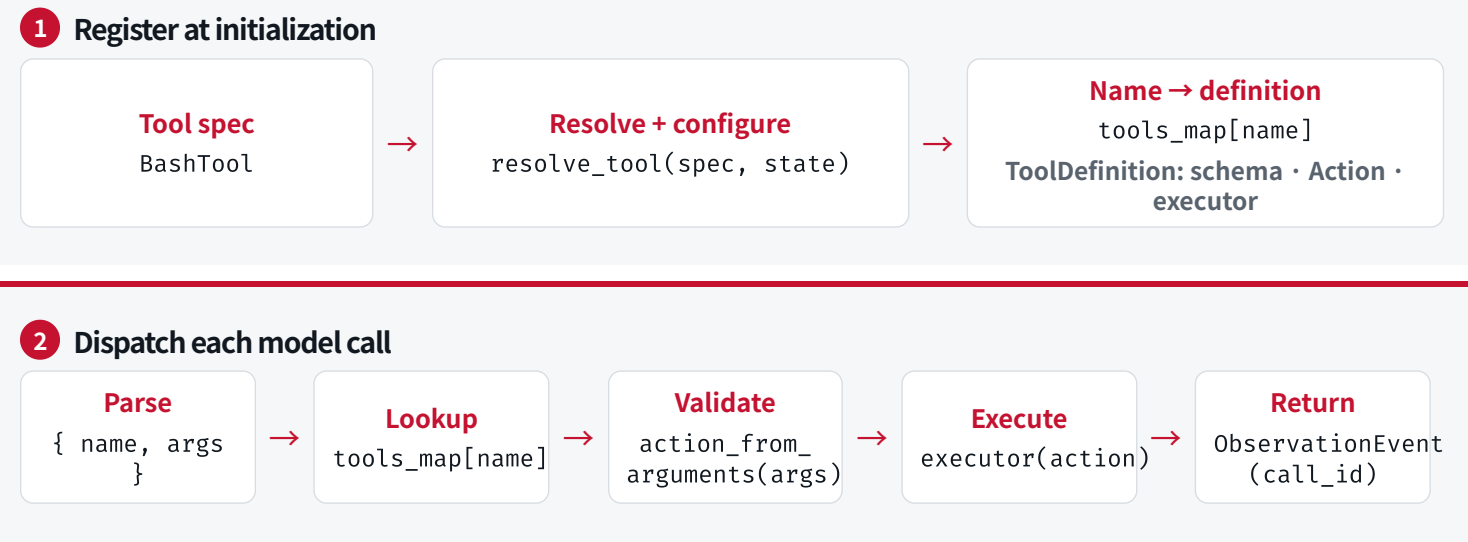
相同工具调用，不同模型协议

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

How OpenHands dispatches a tool call



1 · 初始化时注册：工具规格（如 `BashTool`）→ `resolve\_tool(spec, state)` 解析并配置 → 名称映射到定义 `tools\_map[name]`，得到包含 Schema、Action 和执行器的 `ToolDefinition`。

2 · 分派每次模型调用：解析 `{ name, args }` → 在 `tools\_map[name]` 中查找 → `action\_from\_arguments(args)` 验证 → `executor(action)` 执行 → 返回带 `call\_id` 的 `ObservationEvent`。

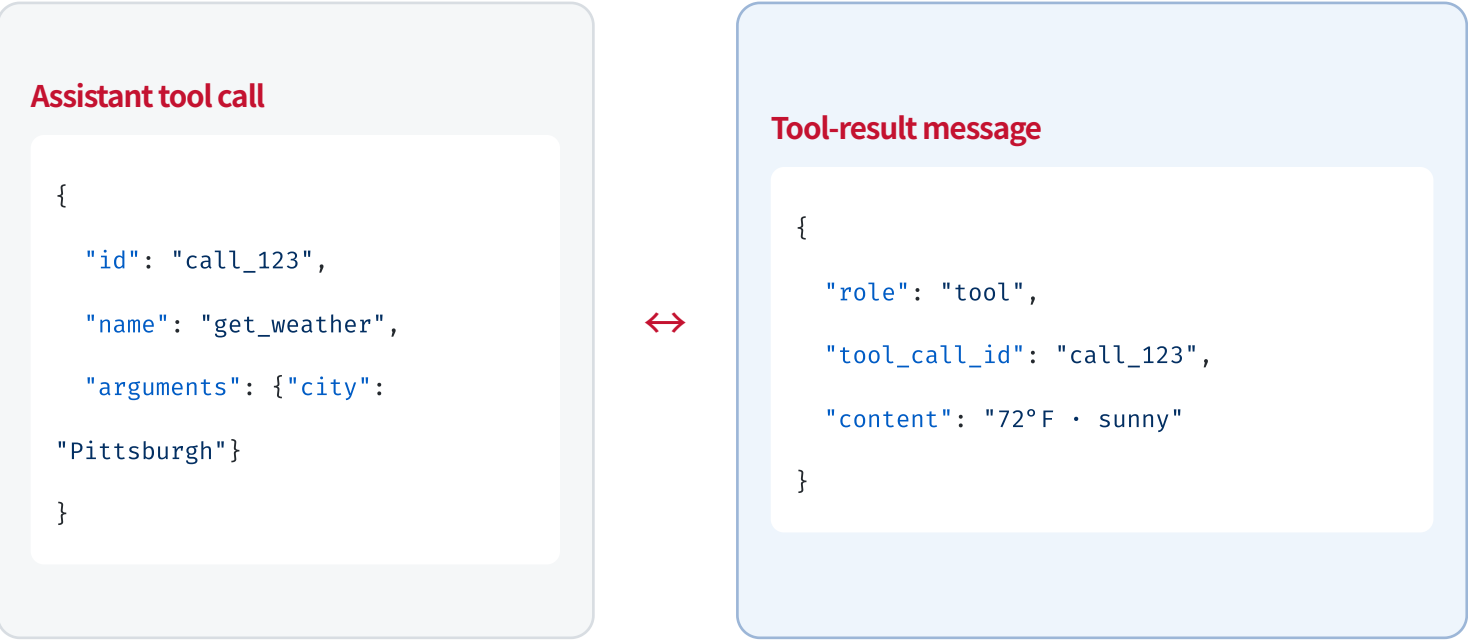
OpenHands 如何分派工具调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Results return by call ID



25

assistant 工具调用：`id: "call\_123"`、`name: "get\_weather"`、参数
`{"city": "Pittsburgh"}`。

工具结果消息：`role: "tool"`、`tool\_call\_id: "call\_123"`、内容 `72°F · sunny`。

`tool\_call\_id` 将结果与调用匹配，包括并行调用。

结果按调用 ID 返回

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



约束解码

约束解码

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Malformed tool calls

Model output

```
{ "name": "get_weather",  
  "arguments": { "city": "Pittsburgh" }
```



Missing }

The harness cannot parse it.

27

模型输出：`{"name":"get\_weather", "arguments":{"city":"Pittsburgh"}}`，缺少`}`。

执行框架无法解析该调用。

格式错误的工具调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Tool-call constraints

Syntax	Shape	Types	Values
valid JSON	expected fields	city is a string	units is C or F

Constraints ensure form—not truth, permissions, tool choice, or task success.

28

语法：有效 JSON。

结构：符合预期字段。

类型：`city` 是字符串。

取值：`units` 为 C 或 F。

约束保证形式正确，但不保证真实性、权限、工具选择或任务成功。

工具调用约束

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

JSON Schema

- Vocabulary for describing JSON data
 - Types, required fields, allowed values
 - Machine-readable validation rules
- ```
{ "type": "object", "properties": { "city": {"type": "string"}, "units": {"enum": ["C", "F"]} }, "required": ["city"], "additionalProperties": false}
```

[JSON Schema Core · Draft 2020-12 ↗](#)

29

用于描述 JSON 数据的词汇体系：类型、必需字段、允许的取值，以及机器可读的验证规则。

示例：`type: object`；`city` 为字符串；`units` 的枚举为 `C`、`F`；`required: ["city"]`；`additionalProperties: false`。

[JSON Schema Core · Draft 2020-12 ↗](#)

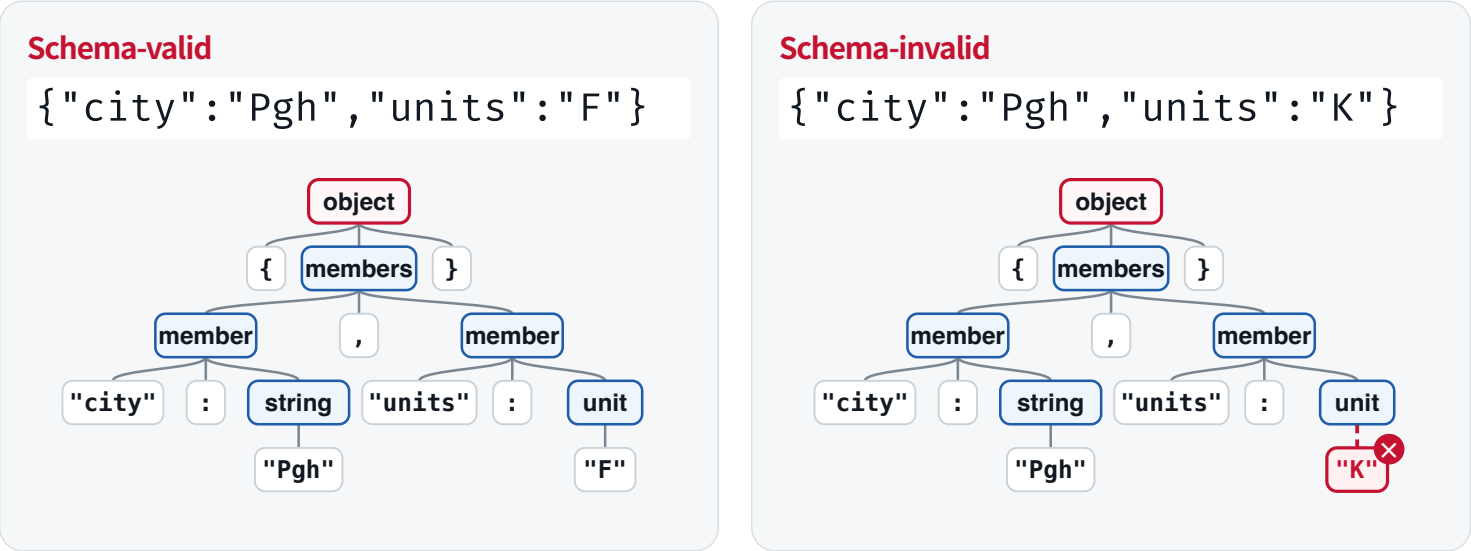
## JSON Schema

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Schema-valid vs. schema-invalid



Both are JSON; only "F" derives from unit → "C" | "F" .

30

符合 Schema 与不符合 Schema

符合 Schema: ``{"city":"Pgh","units":"F"}``。不符合 Schema: ``{"city":"Pgh","units":"K"}``。

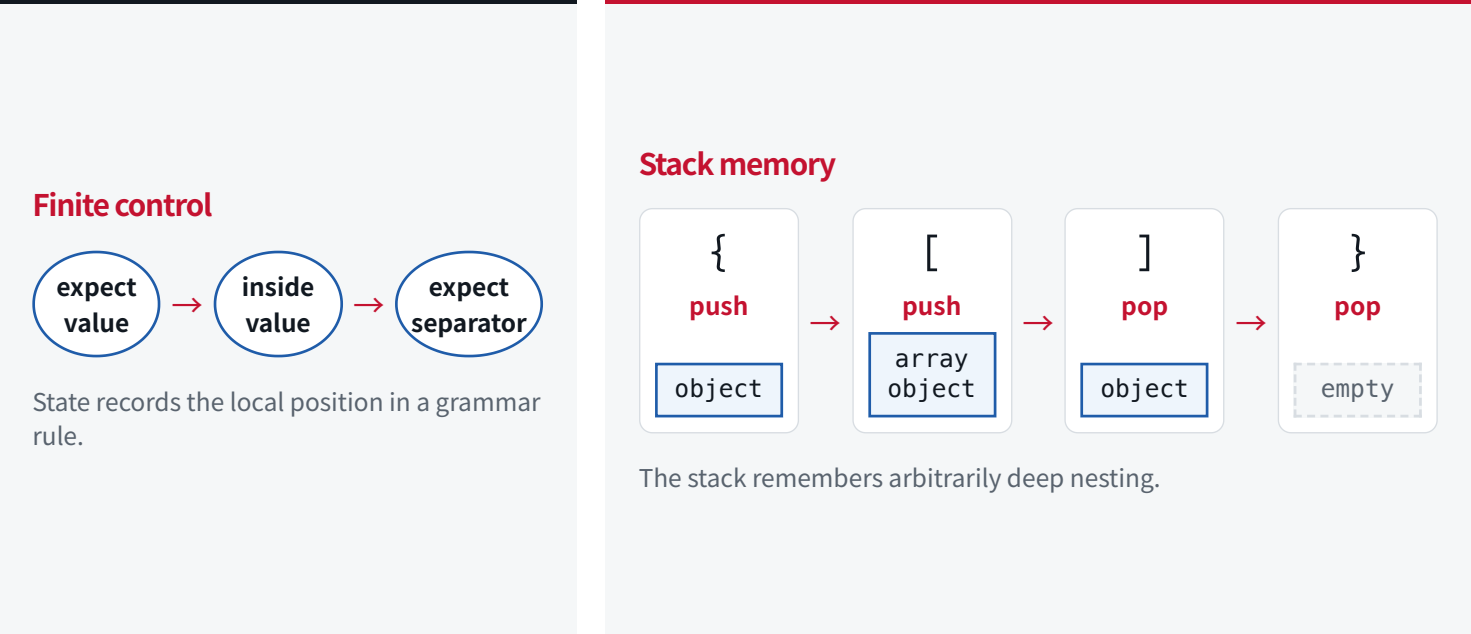
二者都是 JSON；只有 ``F`` 符合取值集合 ``C | F``。

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

A pushdown automaton adds a stack



State tracks the current rule; the stack remembers where nested rules return.

31

有限状态控制：在规则中记录当前局部位置，例如期望值 → 位于值内部 → 期望分隔符。

栈记忆：对 `[ ]` 等嵌套结构执行压栈和弹栈；可记住任意深度的嵌套。

状态跟踪当前规则；栈记住嵌套规则返回的位置。

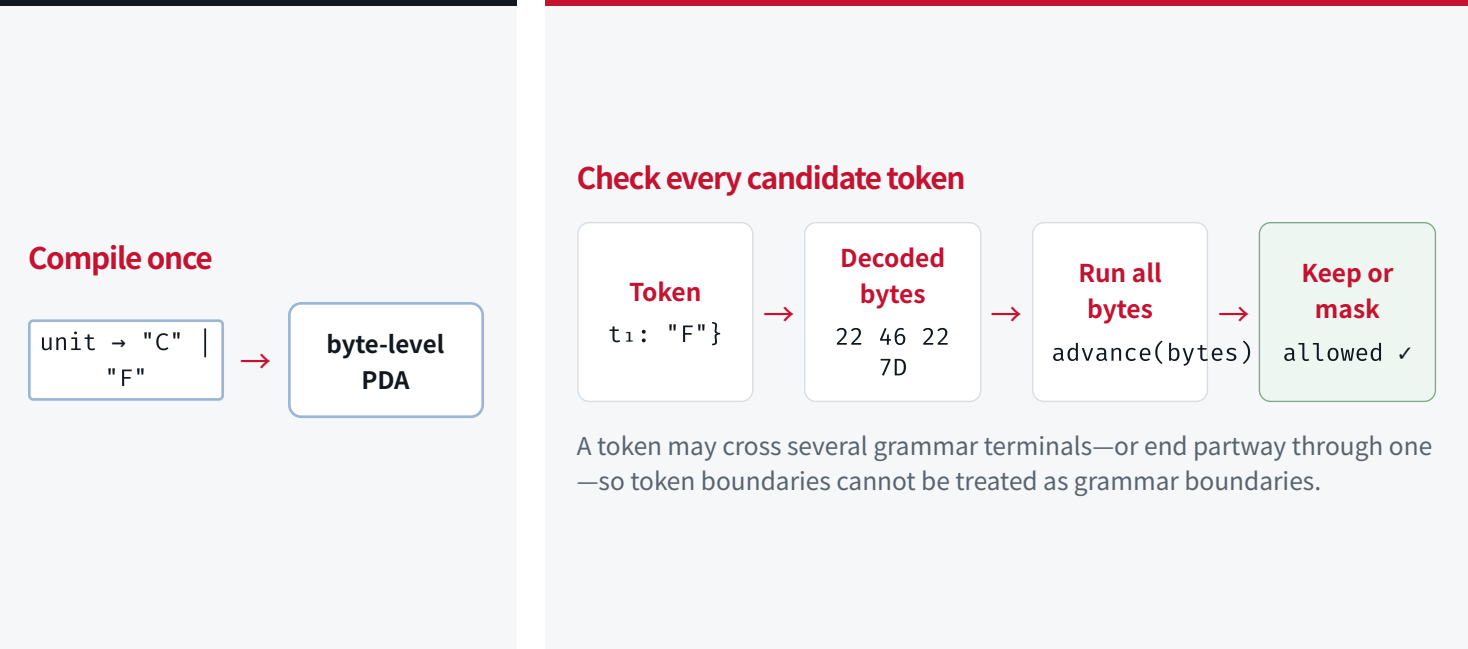
下推自动机增加了栈

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Grammar checking happens below tokenization



XGrammar · Dong et al., MLSys 2025 ↗

32

先编译一次规则 `unit → "C" | "F"`，得到字节级 PDA。对每个候选词元检查：解码词元字节，运行字节并调用 `advance(bytes)`，保留允许的词元。

示例词元 `t1: "F"` 解码为字节 `22 46 22 7D`；结果为允许。

一个词元可能跨越多个语法终结符，也可能只到达某个终结符的中途，因此不能把词元边界当作语法边界。

XGrammar · Dong 等, MLSys 2025 ↗

语法检查发生在词元化之后的更低层

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Example: cached token-mask generation

**1 · Parse the prefix**

Advance persistent PDA stacks over the bytes already emitted.

**2 · Fetch cached decisions**

The stack-top node indexes precomputed validity for context-independent tokens.

**3 · Check the minority**

Run context-dependent candidate tokens against the full stack; union results across matching stacks.

**4 · Mask and continue**

Combine the decisions, mask the logits, sample a token, and update the PDA stacks.

XGrammar · Dong et al., MLSys 2025 ↗

33

- 1 · 解析前缀：在已输出的字节上推进持久化 PDA 栈。
- 2 · 获取缓存决策：栈顶节点为与上下文无关的词元索引预先计算的有效性结果。
- 3 · 检查少数候选：对依赖上下文的候选词元，使用完整栈运行检查；合并匹配栈上的结果。
- 4 · 掩码并继续：合并决策、屏蔽 logits、采样词元，并更新 PDA 栈。

XGrammar · Dong 等, MLSys 2025 ↗

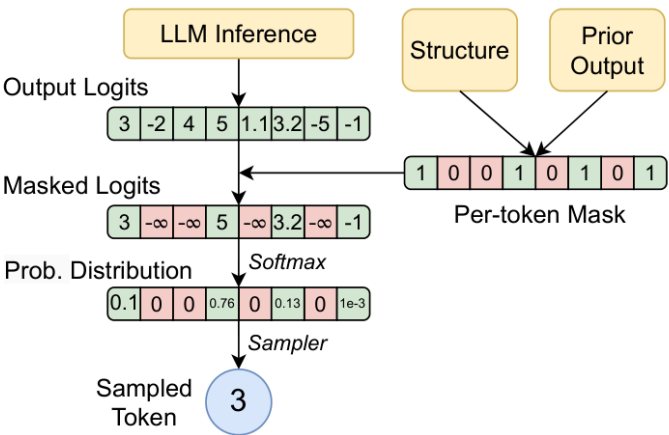
示例：缓存的词元掩码生成

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Token masking



LLM 推理的输出 logits 经逐词元掩码限制后得到概率分布，再由采样器生成所采样的词元 `3`。掩码由结构和先前输出共同决定。

标签：输出 logits、掩码 logits、概率分布、采样器、采样词元、结构、先前输出、逐词元掩码。

XGrammar · Dong 等, MLSys 2025 ↗

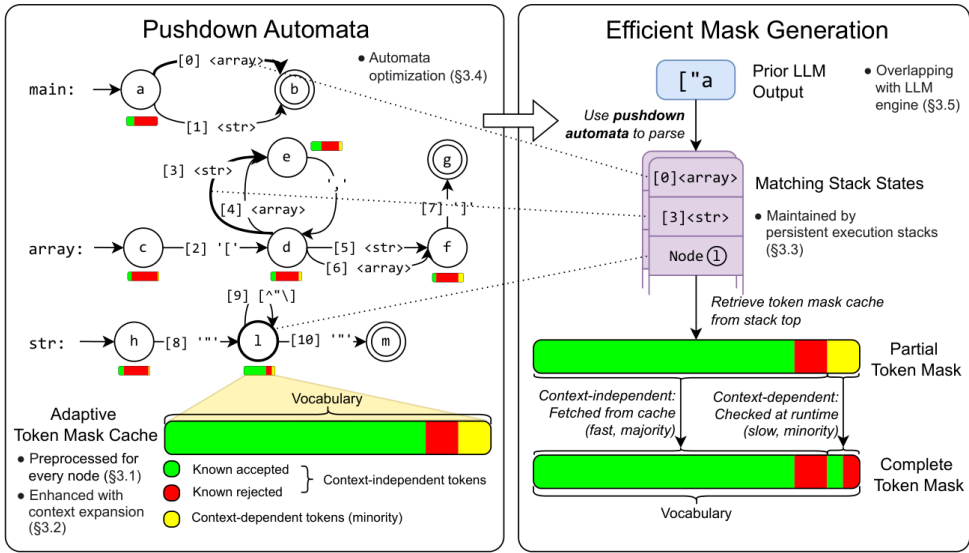
词元掩码

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Case study: where the speedup comes from



Caching handles most tokens; persistent stacks support branching; CPU work overlaps GPU inference.

[XGrammar · Dong et al., MLSys 2025 ↗](#)

下推自动机：针对 ``main``、``array``、``str`` 等语法状态追踪解析分支。自适应词元掩码缓存：对每个节点预处理，并通过上下文扩展增强；已知接受、已知拒绝和依赖上下文的词元分别标示。

高效掩码生成：使用下推自动机解析；按持久化执行栈维护匹配栈状态；从栈顶检索词元掩码缓存。上下文无关候选（多数）从缓存快速获取；依赖上下文的少数候选在运行时检查。由此先得到部分掩码，再形成完整词元掩码。

缓存处理大多数词元；持久化栈支持分支；CPU 工作与 GPU 推理重叠。

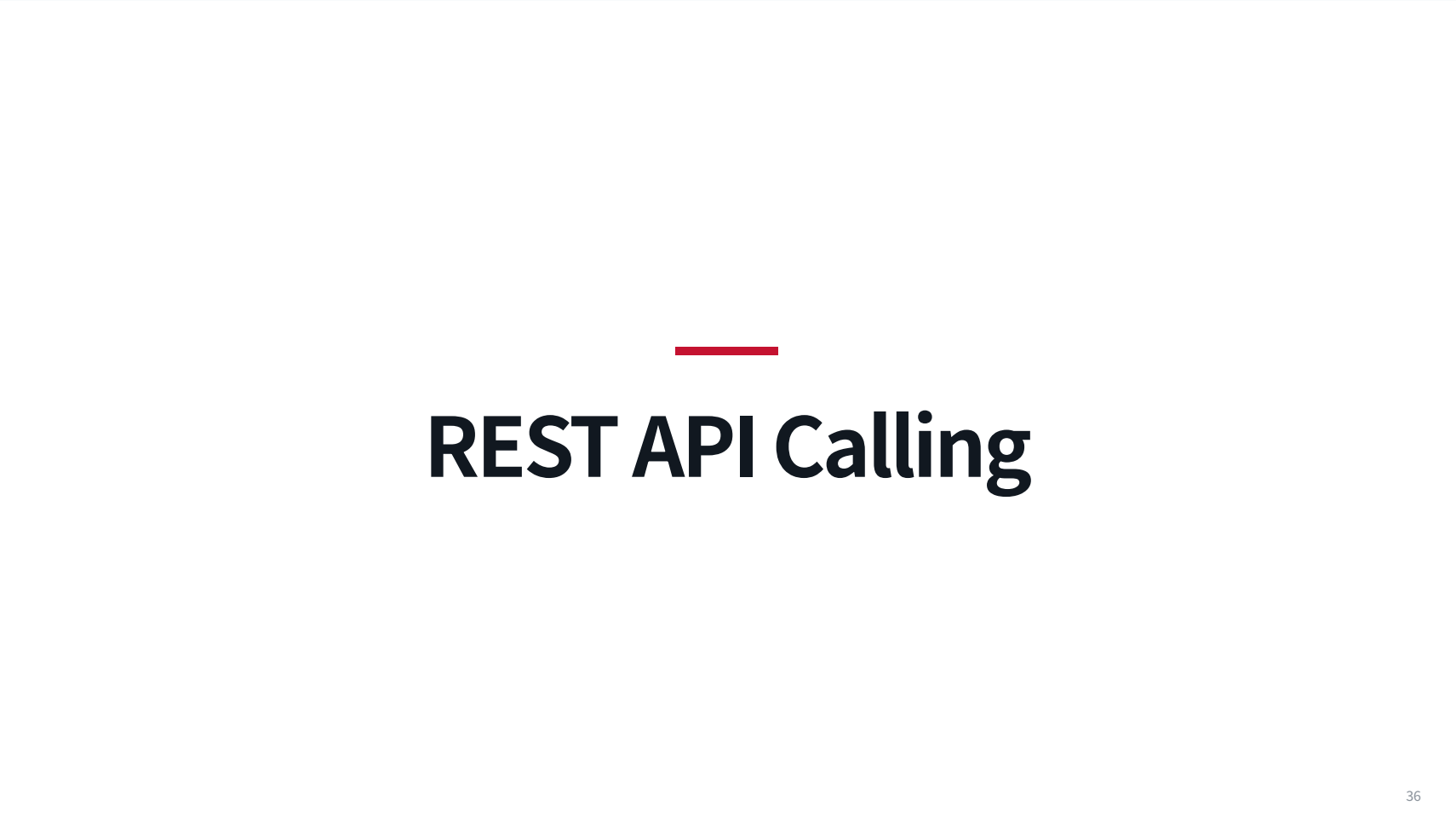
XGrammar · Dong 等, MLSys 2025 ↗

案例研究：加速从何而来

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



REST API 调用

REST API 调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Libraries expose in-process APIs

```
weather_lib.py
from typing import Literal
from pydantic import BaseModel

class Weather(BaseModel):
 summary: str
 temperature: float

def get_weather(
 city: str,
 units: Literal["C", "F"] = "F",
) -> Weather:
 return provider.lookup(city, units)
```

**Caller and function share one process**

Python supplies the name, arguments, types, and return type.

**Not remotely callable yet**

A network client still needs an address, protocol, serialization, and authentication.

A library API is a typed programming interface—not a network service.

37

Python 库 `weather\_lib.py`：导入 `Literal` 和 `BaseModel`；`Weather` 模型包含 `summary: str`、`temperature: float`；`get\_weather(city: str, units: Literal["C", "F"] = "F") -> Weather` 返回 `provider.lookup(city, units)`。

调用方和函数共享同一个进程；Python 提供名称、参数、类型和返回类型。

此时还不能远程调用：网络客户端仍需要地址、协议、序列化和身份验证。

库 API 是带类型的编程接口，不是网络服务。

## 库暴露进程内 API

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

REST APIs expose functions over HTTP

```
api.py
from typing import Annotated, Literal
from fastapi import FastAPI, Security
from fastapi.security import APIKeyHeader
from weather_lib import Weather, get_weather

app = FastAPI(title="Weather API")
key_header = APIKeyHeader(name="X-API-Key")

@app.get("/weather", operation_id="get_weather")
def weather(
 city: str,
 key: Annotated[str, Security(key_header)],
 units: Literal["C", "F"] = "F",
) -> Weather:
 verify(key)
 return get_weather(city, units)
```

**GET /weather**

city=Pittsburgh

units=F

Header: X-API-Key

Response: Weather as JSON

FastAPI binds the library function to a REST endpoint and documents its API-key header.

[FastAPI security and OpenAPI ↗](#)

38

FastAPI 将库函数绑定到 REST 端点，并记录 API key 请求头。示例 `api.py` 导入 `Annotated`、`Literal`、`FastAPI`、`Security`、`APIKeyHeader`，以及 `Weather`、`get\_weather`。

`app = FastAPI(title="Weather API")`；`key\_header = APIKeyHeader(name="X-API-Key")`；端点 `GET /weather` 的 `operation\_id` 为 `get\_weather`。

`weather(city: str, key: Annotated[str, Security(key\_header)], units: Literal["C", "F"] = "F") -> Weather`：验证 key 后调用 `get\_weather(city, units)`。

请求示例：`city=Pittsburgh`、`units=F`；请求头 `X-API-Key`；响应为 JSON 格式的 `Weather`。

FastAPI 安全机制与 OpenAPI ↗

REST API 通过 HTTP 暴露函数

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# REST API servers publish OpenAPI descriptions

```
{
 "paths": {
 "/weather": {
 "get": {
 "operationId": "get_weather",
 "parameters": [
 {
 "name": "city",
 "in": "query",
 "required": true,
 "schema": {
 "type": "string"
 }
 },
 {
 "name": "units",
 "in": "query",
 "schema": {
 "type": "string",
 "enum": ["C", "F"],
 "default": "F"
 }
 }
],
 "security": [
 {
 "APIKeyHeader": []
 }
]
 }
 }
 },
 "components": {
 "securitySchemes": {
 "APIKeyHeader": {
 "type": "apiKey",
 "in": "header",
 "name": "X-API-Key"
 }
 }
 }
}
```

OpenAPI describes the HTTP operation; embedded **JSON Schema** describes its values.

[FastAPI · generated OpenAPI ↗](#)

OpenAPI 文档描述 `/weather` 的 GET 操作，`operationId` 为 `get_weather`。

参数：必需的查询参数 `city`，Schema 类型为字符串；查询参数 `units`，Schema 类型为字符串、枚举为 `C` 和 `F`、默认值为 `F`。

安全要求使用 `APIKeyHeader`；其安全方案类型为 `apiKey`，位置为 `header`，名称为 `X-API-Key`。

OpenAPI 描述 HTTP 操作；其中嵌入的 JSON Schema 描述参数取值。

FastAPI · 自动生成的 OpenAPI ↗

## REST API 服务器发布 OpenAPI 描述

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Coding agents often call REST APIs with curl

```
curl -sS -G https://weather.example/weather \
 --data-urlencode "city=Pittsburgh" \
 --data-urlencode "units=F" \
 -H "X-API-Key: $WEATHER_API_KEY"
```

- 1 · Read the API description**

Select the operation and required parameters.
- 2 · Construct the command**

The shell expands the environment variable; curl sends HTTP.
- 3 · Limit the credential**

A coding agent with shell access may also read or misuse that secret.

Direct REST access is simple, but the agent's shell receives the API credential.

40

命令：`curl -sS -G https://weather.example/weather --data-urlencode "city=Pittsburgh" --data-urlencode "units=F" -H "X-API-Key: \$WEATHER\_API\_KEY"`。

- 1 · 阅读 API 描述：选择操作和必需参数。
- 2 · 构造命令：Shell 展开环境变量，curl 发送 HTTP 请求。
- 3 · 限制凭据：能使用 Shell 的编码智能体也可能读取或滥用该密钥。

直接访问 REST 很简单，但智能体的 Shell 会拿到 API 凭据。

## 编码智能体常用 curl 调用 REST API

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



MCP

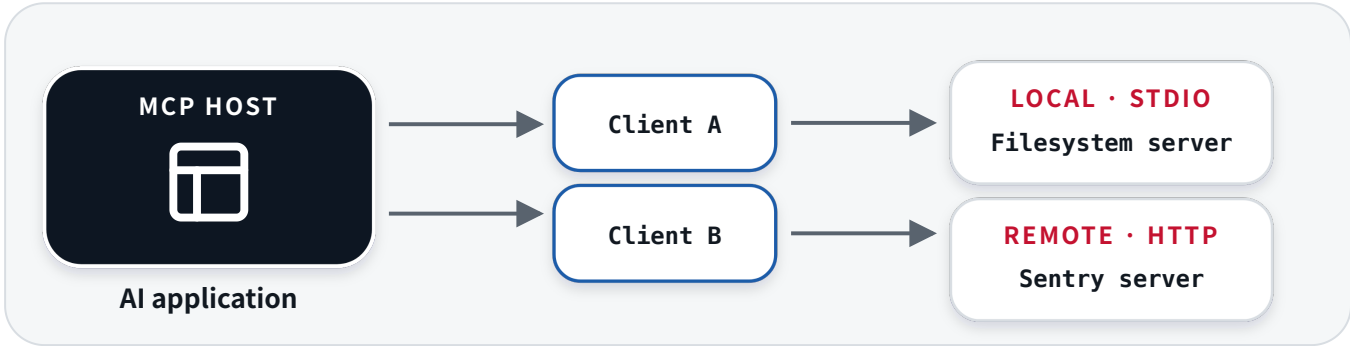
MCP

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

MCP connects a host to servers



MCP is an alternative agent-facing interface—not a requirement for calling REST APIs.

42

MCP 宿主（AI 应用）与客户端 A、客户端 B 相连；本地 · STUDIO：文件系统服务器；远程 · HTTP：Sentry 服务器。

MCP 是面向智能体的另一种接口，并非调用 REST API 的必要条件。

MCP 将宿主连接到服务器

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

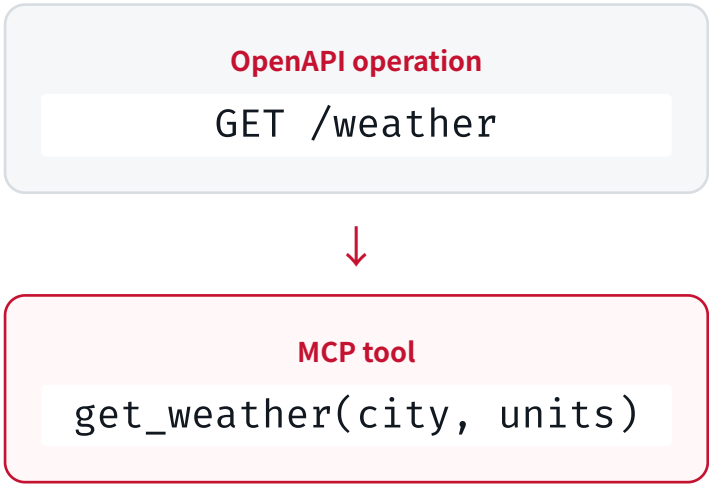
ZH / 中文译文 · 第 1/1 部分

# FastMCP projects OpenAPI operations as MCP tools

```
import httpx2
from fastmcp import FastMCP

spec = httpx2.get(
 "https://weather.example/openapi.json"
).json()

mcp = FastMCP.from_openapi(
 openapi_spec=spec,
 client=httpx2.AsyncClient(
 base_url="https://weather.example"
),
 name="Weather MCP",
)
```



By default, FastMCP converts each OpenAPI route into a discoverable MCP tool.

[FastMCP · OpenAPI integration ↗](#)

43

示例代码使用 `httpx2.get("https://weather.example/openapi.json").json()` 获取规范，再调用 `FastMCP.from_openapi(openapi_spec=spec, client=httpx2.AsyncClient(base_url="https://weather.example"), name="Weather MCP")`。

OpenAPI 操作 `GET /weather` 映射为 MCP 工具 `get_weather(city, units)`。

默认情况下，FastMCP 会将每条 OpenAPI 路由转换为可发现的 MCP 工具。

[FastMCP · OpenAPI 集成 ↗](#)

## FastMCP 将 OpenAPI 操作映射为 MCP 工具

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Start MCP with separate credentials

```
import os, httpx2
from fastmcp import FastMCP
from fastmcp.server.auth.providers.jwt import StaticTokenVerifier

upstream = httpx2.AsyncClient(
 base_url="https://weather.example",
 headers={"X-API-Key": os.environ["UPSTREAM_API_KEY"]})
mcp_auth = StaticTokenVerifier(tokens={
 os.environ["MCP_API_KEY"]: {"client_id": "course-agent"}})

mcp = FastMCP.from_openapi(
 openapi_spec=spec, client=upstream,
 name="Weather MCP", auth=mcp_auth)
mcp.run(transport="http", port=8001)
```

UPSTREAM\_API\_KEY authorizes API calls; MCP\_API\_KEY protects the MCP endpoint.

Static tokens are for teaching/development; production deployments should verify JWTs or use OAuth.

[FastMCP · token verification ↗](#)

上游客户端使用 `UPSTREAM\_API\_KEY` 调用天气 API；MCP 端点由 `MCP\_API\_KEY` 保护。

代码创建 `httpx2.AsyncClient`，设置 `base\_url="https://weather.example"` 和请求头 `X-API-Key: os.environ["UPSTREAM\_API\_KEY"]`；再用 `StaticTokenVerifier` 校验 `MCP\_API\_KEY`，其客户端 ID 为 `course-agent`。

使用 `FastMCP.from\_openapi(...)` 建立名为 `Weather MCP` 的服务器，并以 `transport="http"`、`port=8001` 运行。

静态 token 适用于教学 / 开发；生产部署应验证 JWT 或使用 OAuth。

FastMCP · token 验证 ↗

## 使用不同凭据启动 MCP

第 2 讲 · 智能体能力 1：工具使用

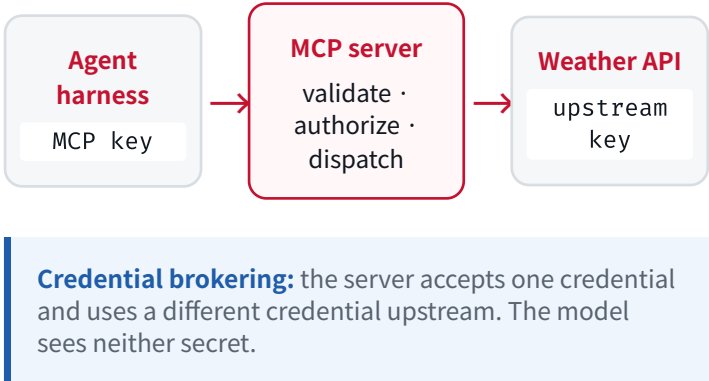
EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

The MCP client uses its own API key

```
import os
from fastmcp import Client

async with Client(
 "https://mcp.example/mcp",
 auth=os.environ["MCP_API_KEY"],
) as client:
 tools = await client.list_tools()
 result = await client.call_tool(
 "get_weather",
 {"city": "Pittsburgh", "units": "F"},
)
```



Separate credentials let the MCP server mediate access instead of forwarding its upstream secret.

[FastMCP · bearer client authentication ↗](#)

Agent harness 连接 MCP 服务器，再连接 Weather API。客户端使用 ``Client("https://mcp.example/mcp", auth=os.environ["MCP_API_KEY"]);``；调用 ``list_tools()``，再用 ``call_tool("get_weather", {"city":"Pittsburgh", "units":"F"})``。

服务器验证和授权 MCP key，再使用上游 key 分派请求。凭据代理：服务器接受一种凭据，并用另一种凭据访问上游；模型看不到任一密钥。

不同凭据让 MCP 服务器能够中介访问，而不是转发上游密钥。

FastMCP · bearer 客户端身份验证 ↗

MCP 客户端使用自己的 API key

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

# REST APIs and MCP: shared schemas, different contracts

|                | OpenAPI / HTTP API                                         | MCP                                                    |
|----------------|------------------------------------------------------------|--------------------------------------------------------|
| Shared         | Names, descriptions, and JSON Schema for structured inputs |                                                        |
| Discovery      | Fetch an OpenAPI document                                  | Call <code>tools/list</code> at runtime                |
| Invocation     | HTTP verb + path + parameters                              | <code>tools/call</code> over an MCP transport          |
| Authentication | Client authenticates to the API                            | Client authenticates to MCP; upstream auth is separate |
| Scope          | Describes HTTP operations                                  | Also supports resources, prompts, and extensions       |

OpenAPI describes a web API; MCP standardizes how an AI host discovers and invokes capabilities.

46

ZH / 中文译文 · 第 1/1 部分

共同点：名称、描述以及结构化输入的 JSON Schema。

发现：获取 OpenAPI 文档；MCP 则在运行时调用 `tools/list`。

调用：HTTP 方法 + 路径 + 参数；MCP 则通过 MCP 传输调用 `tools/call`。

身份验证：客户端向 API 验证身份；客户端向 MCP 验证身份，上游身份验证另行处理。

范围：OpenAPI 描述 HTTP 操作；MCP 还支持资源、提示和扩展。

OpenAPI 描述 Web API；MCP 标准化 AI 宿主发现和调用能力的方式。

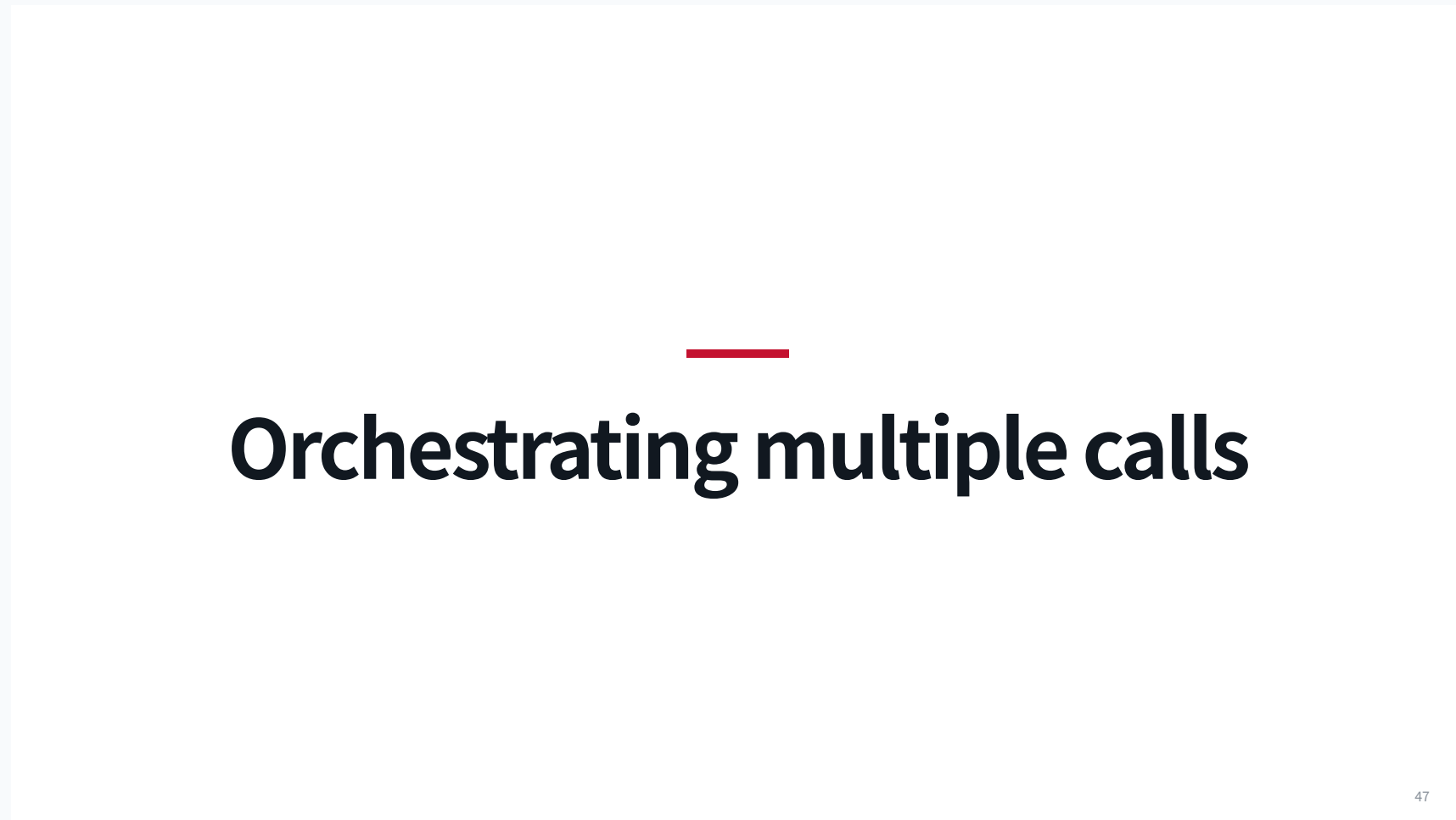
## REST API 与 MCP：共用 Schema，不同契约

## 第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

## 编排多个调用



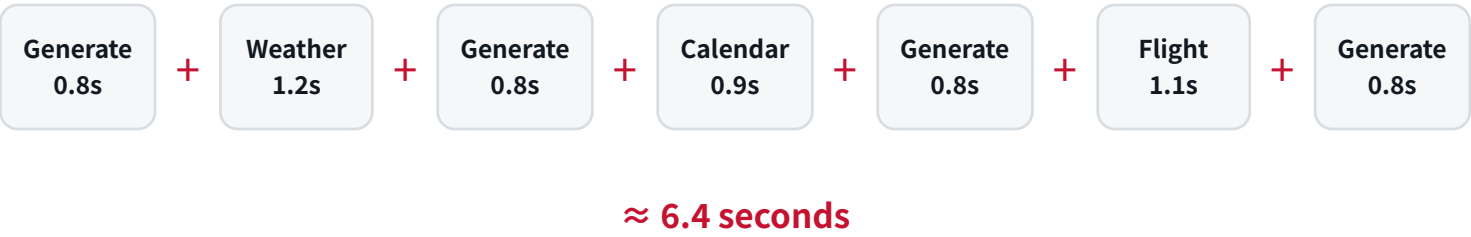
## 编排多个调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Serial tool-call cost



生成 0.8 秒 + 天气 1.2 秒 + 生成 0.8 秒 + 日历 0.9 秒 + 生成 0.8 秒 + 航班 1.1 秒 + 生成 0.8 秒 ≈ 6.4 秒。

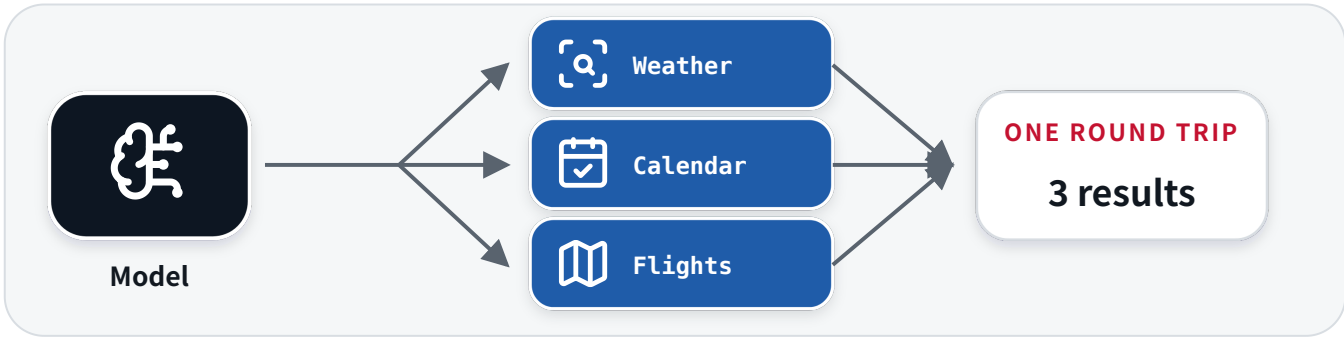
串行工具调用的成本

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Parallel tool calls



Independent calls can share one generation and one wait.

Two generation phases plus the slowest tool:  $2 \times 0.8 + \max(1.2, 0.9, 1.1) \approx \mathbf{2.8 \text{ seconds}}$ .

49

天气、日历、航班三个调用并行发出，一次往返得到 3 个结果。

独立调用可以共用一次生成和一次等待。

两阶段生成加上最慢工具耗时：` $2 \times 0.8 + \max(1.2, 0.9, 1.1) \approx 2.8` 秒。$

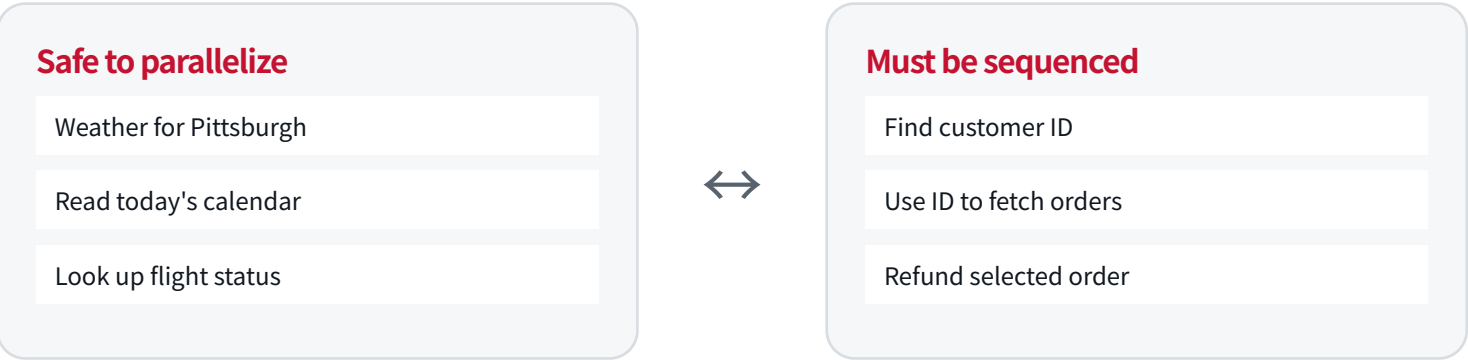
并行工具调用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Parallelize only without dependencies



Parallelize only independent, safely concurrent calls.

50

可安全并行：查询匹兹堡天气；读取今天的日历；查询航班状态。

必须按顺序执行：查找客户 ID → 使用该 ID 获取订单 → 退还所选订单。

只并行执行彼此独立、且可以安全并发的调用。

只有在没有依赖时才并行

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Concurrent execution

```
async def execute(call):
 args = json.loads(call.arguments)
 value = await TOOLS[call.name](**args)
 return call.call_id, value

results = await asyncio.gather(
 *(execute(call) for call in function_calls),
 return_exceptions=True,
)
```

Preserve call IDs when results finish out of order.

``execute(call)``: 用 ``json.loads(call.arguments)`` 解析参数; 执行 ``await TOOLS[call.name](**args)``; 返回 ``(call.call_id, value)``。

使用 ``asyncio.gather(*(execute(call) for call in function_calls), return_exceptions=True)`` 并发执行。

即使结果按不同顺序返回，也要保留调用 ID。

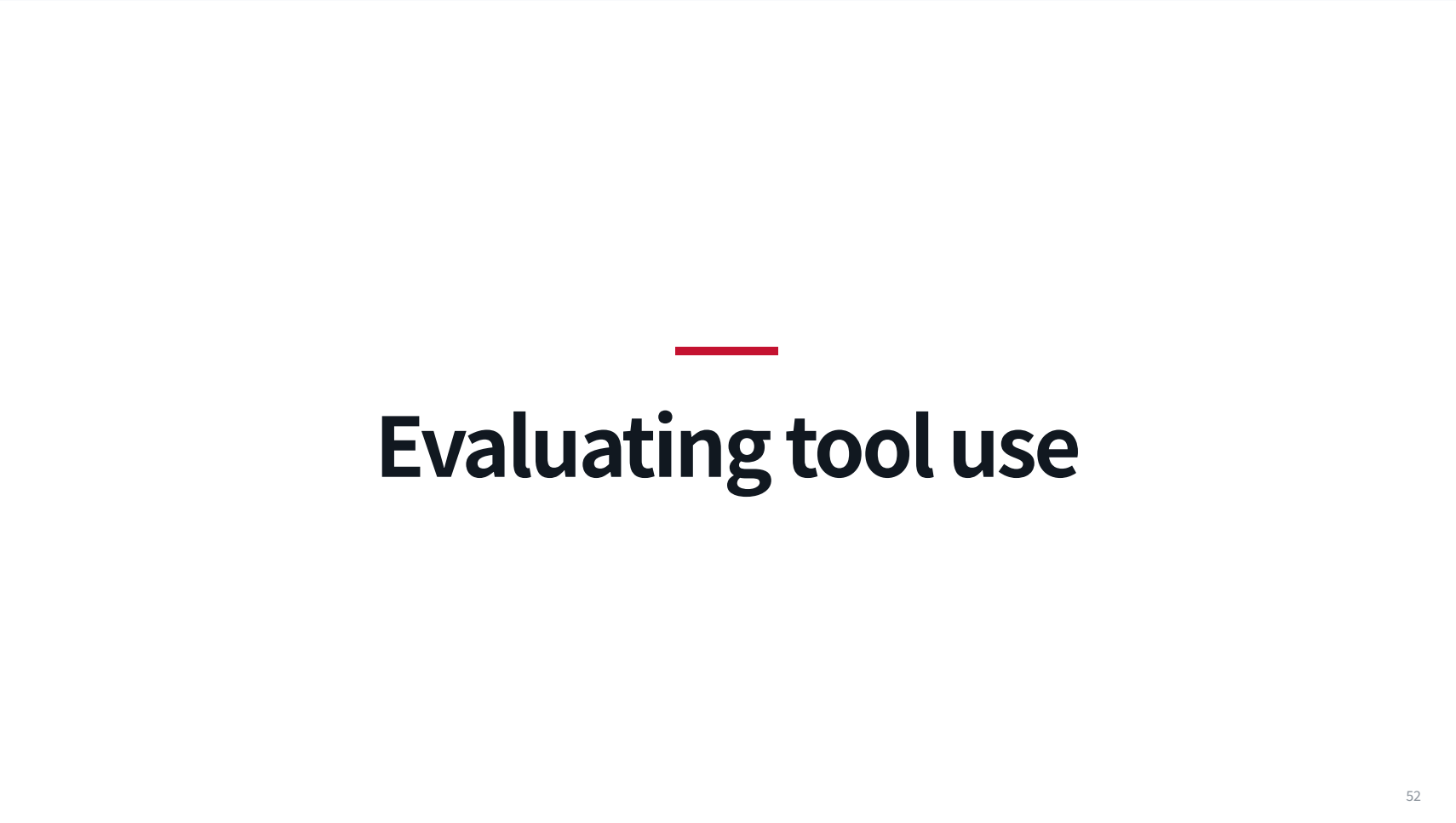
## 并发执行

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

评估工具使用



评估工具使用

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Berkeley Function Calling Leaderboard

|                                                                        |                                                                                 |
|------------------------------------------------------------------------|---------------------------------------------------------------------------------|
| <b>Single turn</b><br>simple · multiple · parallel · multiple-parallel | <b>Multi-turn</b><br>base · missing function · missing parameter · long context |
| <b>Agentic</b><br>web search · memory                                  | <b>Robustness</b><br>hallucination measurement · format sensitivity             |

[Berkeley Function-Calling Leaderboard V4 ↗](#)

53

单轮：简单、多工具、并行、多工具并行。

多轮：基础、缺少函数、缺少参数、长上下文。

智能体能力：网页搜索、记忆。

稳健性：幻觉测量、格式敏感性。

[Berkeley Function-Calling Leaderboard V4 ↗](#)

## 伯克利函数调用排行榜

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

# Evaluate the whole stack

| Level      | Question       | Example metric     | Execution? | Typical failure        |
|------------|----------------|--------------------|------------|------------------------|
| Selection  | Right tool?    | Precision / recall | No         | Missing or extra call  |
| Arguments  | Right values?  | AST / schema match | No         | Wrong field or value   |
| Trajectory | Right order?   | Sequence success   | Usually    | Bad dependency         |
| Task       | Goal achieved? | End-to-end success | Yes        | Plausible wrong answer |

54

## 评估整个技术栈

ZH / 中文译文 · 第 1/1 部分

选择：工具选对了吗？指标示例：准确率 / 召回率；是否执行：否；典型失败：漏调用或多调用。

参数：取值正确吗？指标示例：AST / Schema 匹配；是否执行：否；典型失败：字段或取值错误。

轨迹：顺序正确吗？指标示例：序列成功率；通常会执行；典型失败：依赖关系错误。

任务：是否达成目标？指标示例：端到端成功率；会执行；典型失败：答案看似合理但实际错误。

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Quality has operational dimensions

|                                                                                                                         |                                                                                                                             |                                                                                                                    |
|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|
| <b>Efficiency</b><br><b>Latency · calls · tokens · cost</b><br>A correct agent can still be unusably slow or expensive. | <b>Reliability</b><br><b>Timeouts · retries · partial failure</b><br>Report task success under realistic provider behavior. | <b>Safety</b><br><b>Policy · permissions · side effects</b><br>Test adversarial outputs and consequential actions. |
|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|

55

效率：延迟、调用次数、词元数、成本。即使智能体正确，也可能慢到无法使用或成本过高。

可靠性：超时、重试、部分失败；应在现实的服务提供方行为下报告任务成功率。

安全：策略、权限、副作用；测试对抗性输出和后果重大的行动。

质量还包括运行层面的维度

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

# Operational tool failures

| Model                                              | Harness                                        | Provider or tool                              |
|----------------------------------------------------|------------------------------------------------|-----------------------------------------------|
| wrong tool · bad arguments ·<br><br>ignores result | parse · auth · validation · ID<br><br>mismatch | timeout · rate limit · execution<br><br>error |

Production evaluation needs benchmark accuracy and operational failure rates.

56

模型：工具选错、参数错误、忽略结果。

执行框架：解析、身份验证、验证、ID 不匹配。

服务提供方或工具：超时、速率限制、执行错误。

生产环境评估既需要基准准确率，也需要运行故障率。

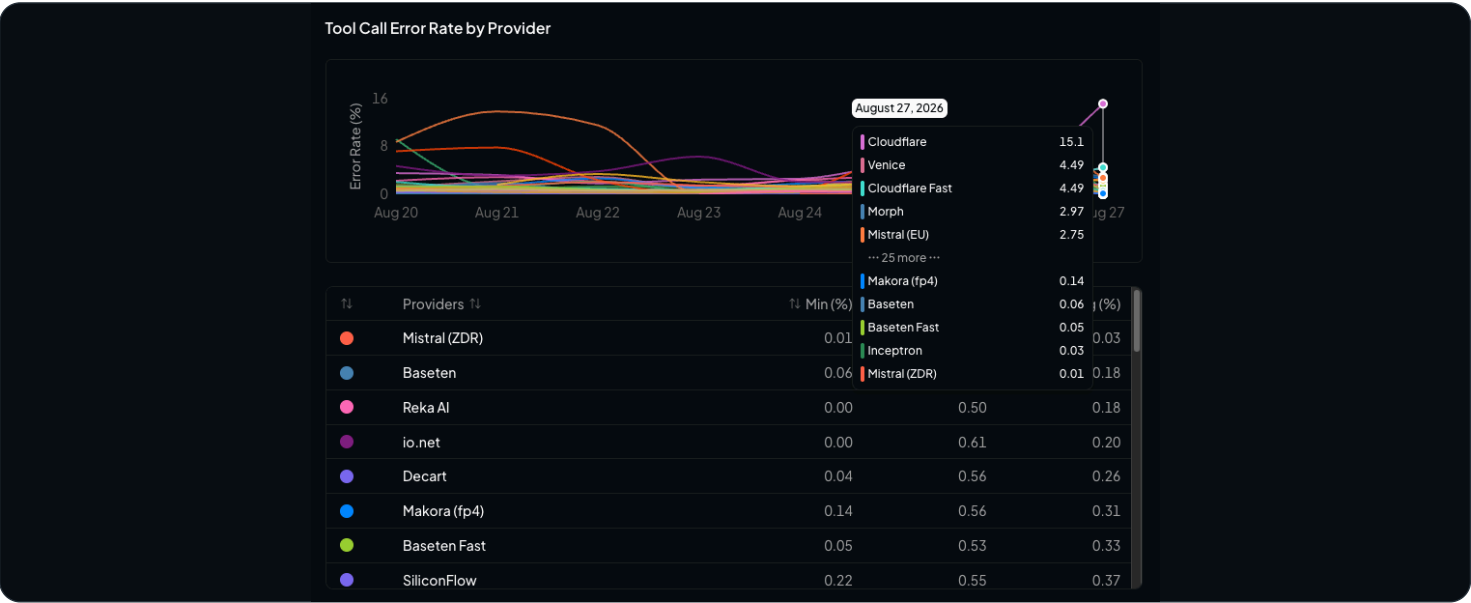
## 运行中的工具故障

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Provider tool-call error rates



OpenRouter provider analytics · Aug. 27, 2026 ↗

57

图表标题：按服务提供方统计的工具调用错误率；纵轴：错误率（%）；横轴日期：8 月 20 日至 8 月 27 日。

2026 年 8 月 27 日可见数据：Cloudflare 15.1；Venice 4.49；Cloudflare Fast 4.49；Morph 2.97；Mistral (EU) 2.75；另有 25 个服务提供方未展开；Makora (fp4) 0.14；Baseten 0.06；Baseten Fast 0.05；Inceptron 0.03；Mistral (ZDR) 0.01。

OpenRouter 服务提供方分析 · 2026 年 8 月 27 日 ↗

译注 / 与原文翻译分开

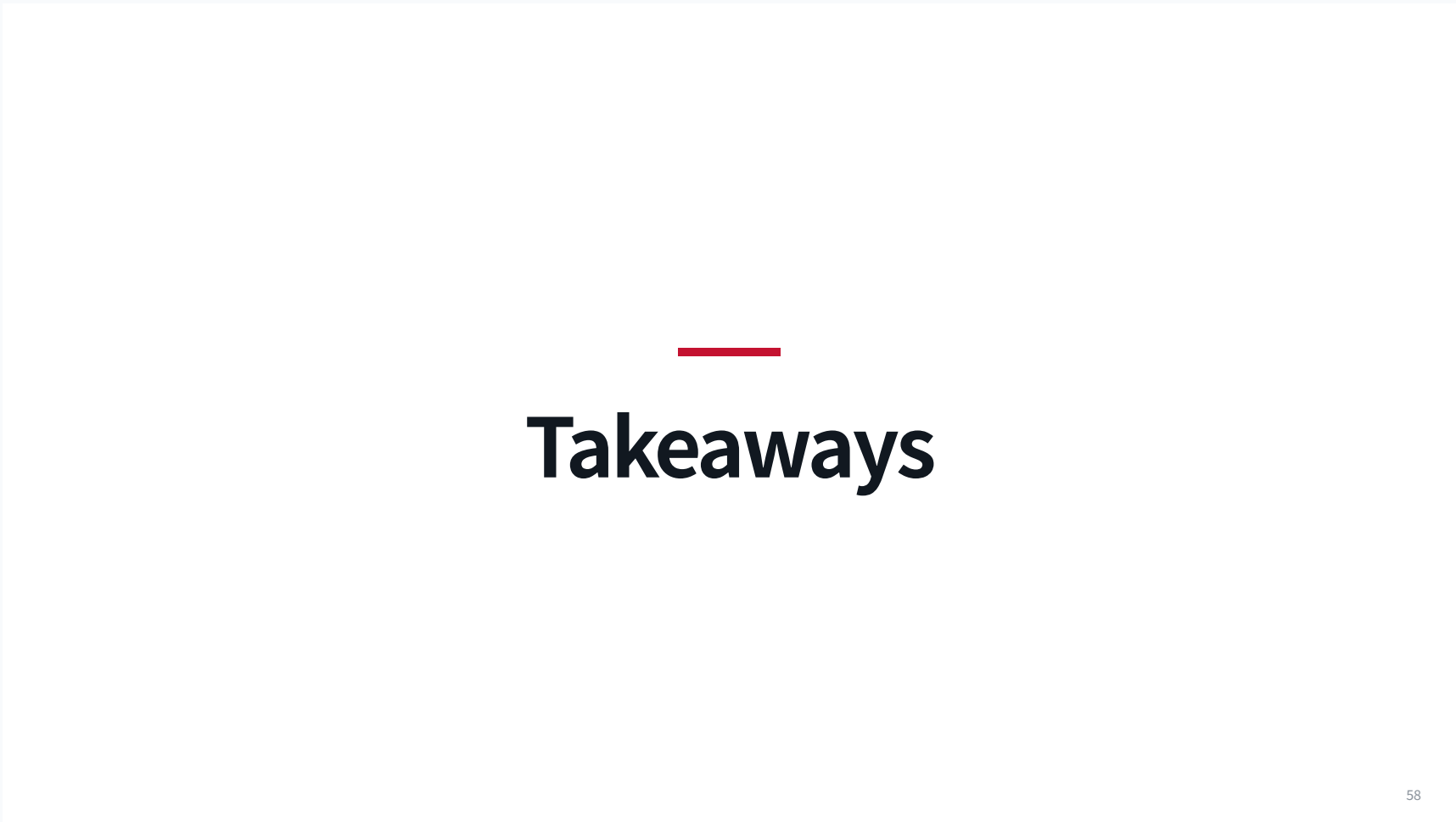
截图中的列表仅展开了部分服务提供方，表格下半部分被裁切；不补写不可见的名称和数值。

服务提供方的工具调用错误率

## 第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分



要点

要点

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

Tool use is a layered system

- 1

**Capabilities:** tools add retrieval, execution, generation, and application actions.
- 2

**Mechanics:** schemas enter the prompt; model protocols emit calls; harnesses validate, dispatch, and match results by ID.
- 3

**Constraints:** JSON Schema and grammar masks guarantee form—not semantic correctness.
- 4

**Interfaces:** direct REST calls and MCP are alternative ways to expose capabilities and manage credentials.
- 5

**Systems:** orchestrate dependencies and concurrency; evaluate selection, arguments, outcomes, cost, and operational failures.

59

- 1 · 能力：工具增加检索、执行、生成和应用操作能力。
- 2 · 机制：Schema 进入提示；模型协议产生调用；执行框架验证、分派调用，并按 ID 匹配结果。
- 3 · 约束：JSON Schema 和语法掩码保证形式正确，而非语义正确。
- 4 · 接口：直接调用 REST 和使用 MCP 是暴露能力、管理凭据的不同方式。
- 5 · 系统：编排依赖和并发；评估工具选择、参数、结果、成本和运行故障。

工具使用是一个分层系统

第 2 讲 · 智能体能力 1：工具使用

EN / 英文官方原页 · 保持原样

ZH / 中文译文 · 第 1/1 部分

下一讲：智能体能力 2——长上下文智能体的上下文管理。



Thank you, Questions?

Next Class: Agent Capabilities 2 — Context Management for Long-Context LLMs

谢谢，欢迎提问