Humanize：

迈向高效可扩展的 Agent Flow 基础设施

张子健 (Zijian Zhang)

CS PhD Student @ Tsinghua University, College of AI

Research Intern @ NVIDIA Research Singapore

Core Developer @ Humanfia

三步演进

从单一循环迈向可扩展的 Flow 基础设施

第一步

RLCR 与 Flame-Chase

Humanize 1

- 从氛围编程到许愿式编程
- RLCR：生成与审查求解约束满足
- Flame-Chase：异构交替求解最优化
- 单一 Flow 的局限与多 Flow 组合

→

第二步

Flow 运行时与 Flow 科学

Humanize 2

- 用原生异步 Python 编排任意 Flow
- 统一驱动多款编程 Agent CLI
- Flowverse 生态与五步快速上手
- Flow 科学：问题归约与系统级消融

→

第三步

可扩展 Flow 基础设施

Humanize NEXT

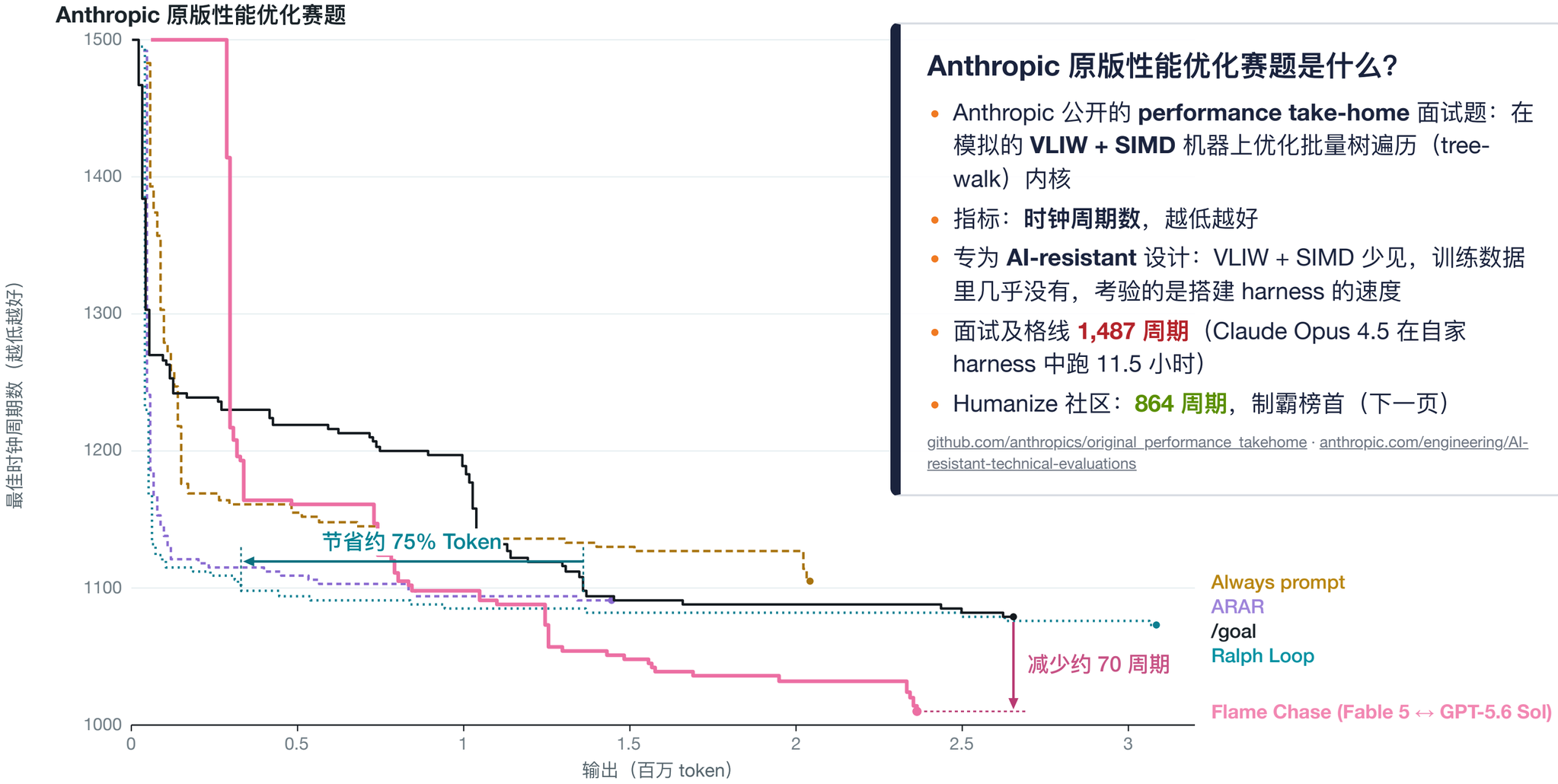
- 从 Flow 科学规律提炼系统需求
- Exomyth：跨层级全局可视化引擎
- CogAnchor：透明远程与容器执行
- 动态 DAG 并发调度与横向扩展

Humanize 主页 humanfia.ai · github.com/humanfia/humanize · docs.humanfia.ai/humanize

NVIDIA

从 Session 到 Flow

面对复杂任务，仅靠简单的 /goal 远远不够



Anthropic 原版性能优化赛题是什么?

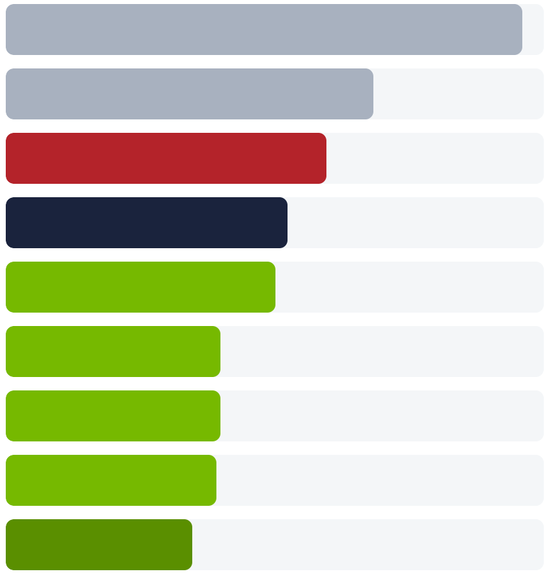
- Anthropic 公开的 **performance take-home** 面试题：在模拟的 **VLIW + SIMD** 机器上优化批量树遍历（tree-walk）内核
- 指标：**时钟周期数**，越低越好
- 专为 **AI-resistant** 设计：VLIW + SIMD 少见，训练数据里几乎没有，考验的是搭建 harness 的速度
- 面试及格线 **1,487 周期**（Claude Opus 4.5 在自家 harness 中跑 11.5 小时）
- Humanize 社区：**864 周期**，制霸榜首（下一页）

github.com/anthropics/original_performance_takehome · anthropic.com/engineering/AI-resistant-technical-evaluations

Humanize 社区制霸 Anthropic 性能赛题榜首

在模拟的 VLIW + SIMD 机器上优化批量树遍历内核 · 时钟周期数，越低越好

性能赛题最佳时钟周期数 ↓ 越低越好



864 周期 = 比 Anthropic 面试及格线（1,487）快 **1.72×**

并列第一的 9 个席位中，Humanize 社区占据 **5 个**

Scoreboard

Without Indices | With Indices

Rank	User	Best Score	Submissions
1	@hyln9	864 cycles	3
2	@josusanmartin	864 cycles	173
3	@antoinegg03	864 cycles	3
4	@xrorrim	864 cycles	5
5	@zhiychen	864 cycles	6
6	@ligeng_zhu	864 cycles	12
7	@justinwetch	864 cycles	12
8	@txdywy	864 cycles	51
9	@alan_wang	864 cycles	9
10	@ryan_kirkman	865 cycles	10
11	@junlin_ai	866 cycles	19
12	@HaydenCC	869 cycles	13

vliw-challenge.fly.dev，2026 年 11 月 7 日 · **绿色**：Humanize 社区

github.com/anthropics/original_performance_takehome · anthropic.com/engineering/AI-resistant-technical-evaluations · vliw-challenge.fly.dev（Without Indices）

NVIDIA

PART 1 · 第一步

Humanize 1：RLCR 与 Flame-Chase

从氛围编程到许愿式编程，从软件开发到算子优化

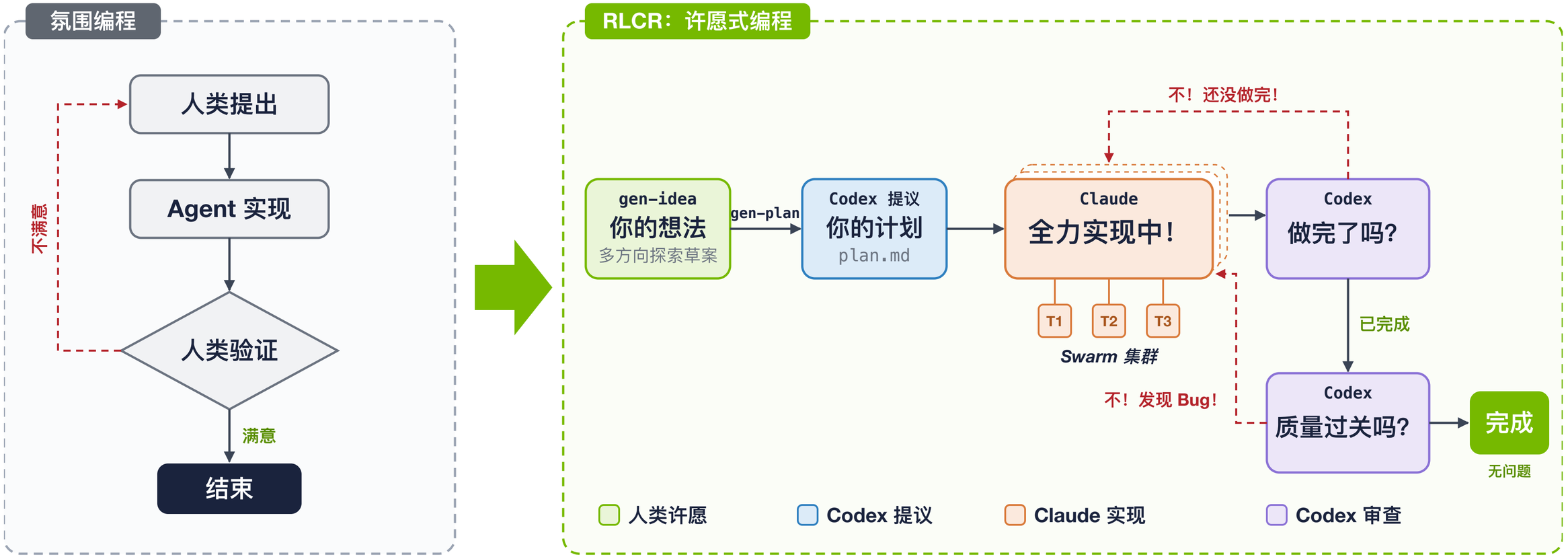
Sihao Liu[1,2], Ligeng Zhu[1], **Zijian Zhang**[1,4], Dongyun Zou[3,5], Zhengyang Zhang[4], Changye Li[4], Song Bian[1], Song Han[1,3], Tony Nowatzki[2]

[1]NVIDIA · [2]UCLA · [3]MIT · [4]Tsinghua University · [5]Nunchux AI

Humanize 判别工程 · arXiv:2610.08900

Humanize 1：带 Codex 审查的 Ralph Loop (RLCR)

从氛围编程到许愿式编程



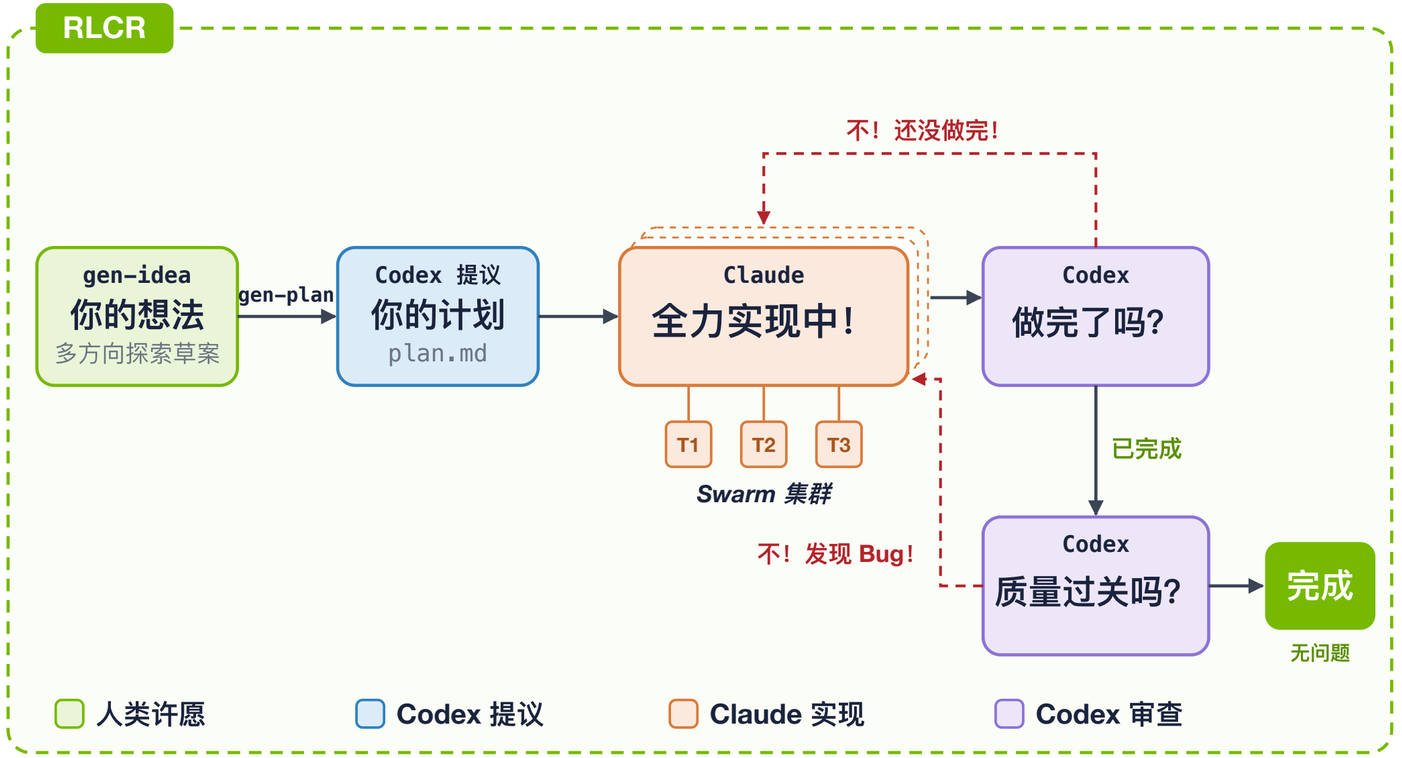
RLCR：人类只需许愿，Codex 提议与审查、Claude 实现，循环推进直至无可挑剔

单一 Flow 无法解决所有问题

没有免费的午餐。

- RLCR 擅长**从零构建完整项目**
- 为什么说编程是**约束满足问题（CSP）**？
  - ✓ 实现所有需求功能
  - ✓ 通过全部测试
  - ✓ 通过静态检查（Lint）
  - ✓ **我们并不关心具体代码细节**

因此我们只需要**一个可行解**

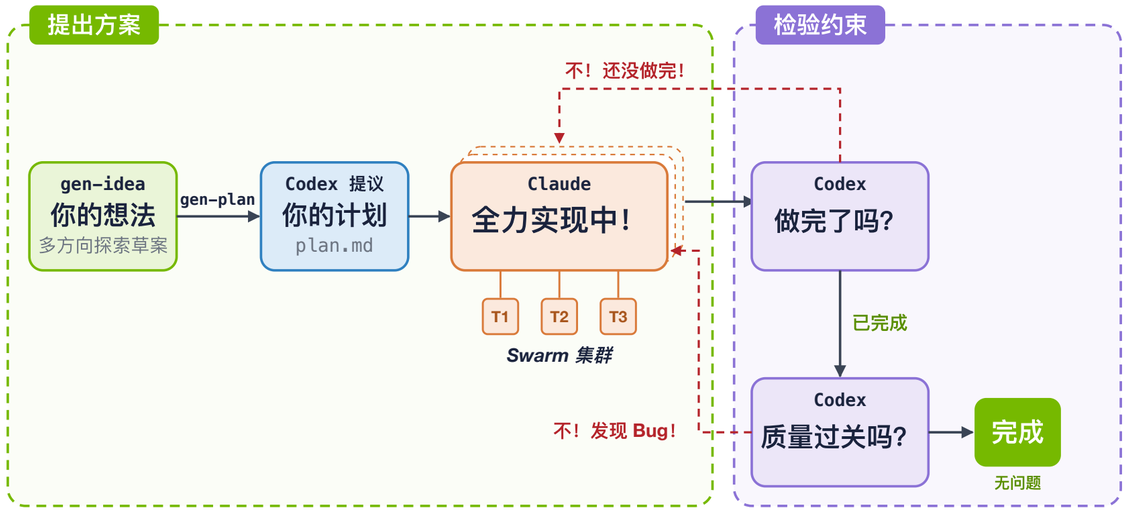


单一 Flow 无法解决所有问题

没有免费的午餐。

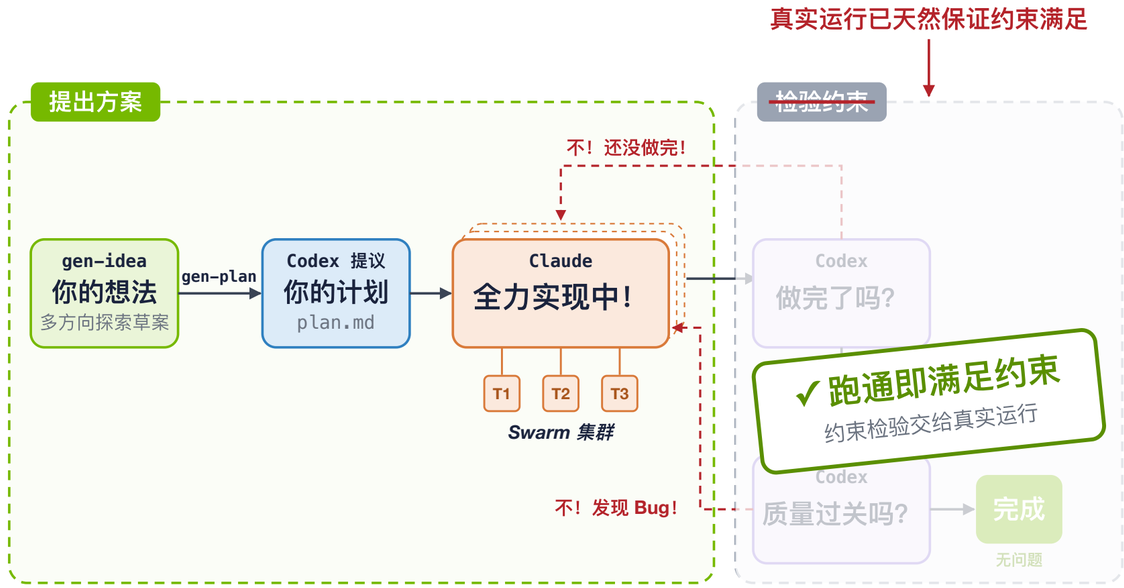
对于 CUDA Kernel，仅仅能跑通远远不够，我们需要的是**极致性能的实现**

工程任务 → 约束满足



Codex 审查把关约束，**只需一个可行解**

算子任务 → 最优化



检验环节已不再是瓶颈，我们**需要最优解**：持续逼近更快的实现

单一 Flow 无法解决所有问题

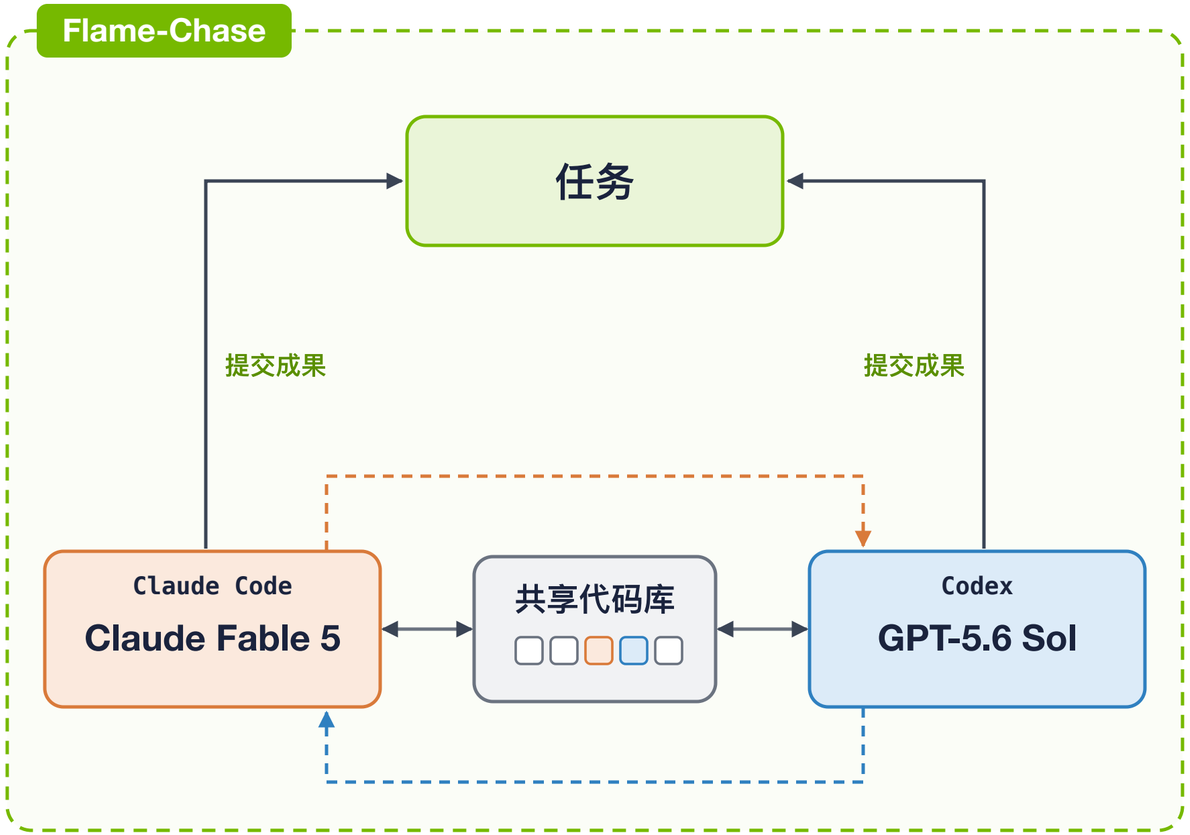
但丰富的 Flow 组合可以。

- 逐火循环（**Flame-Chase**）专为算子设计而生
  - **Claude Code + Claude Fable 5** 偏好大刀阔斧重构，但容易导致性能退化
  - **Codex + GPT-5.6 Sol** 擅长精细调优，但容易陷入局部最优

借助 Flame-Chase，我们能**跳出当前局部最优**，跃迁至更好的局部最优



命名灵感源自《崩坏：星穹铁道》中的「逐火之旅」



单一 Flow 无法解决所有问题

但丰富的 Flow 组合可以。

RLCR

擅长求解

约束满足问题

Flame-Chase

擅长求解

最优化问题

- 面对更加多样化的复杂问题类型与探索范式
- 我们需要一个**更灵活、更强大的框架**，以承载各式各样的 Flow

PART 2 · 第二步

Humanize 2：Flow 运行时与 Flow 科学

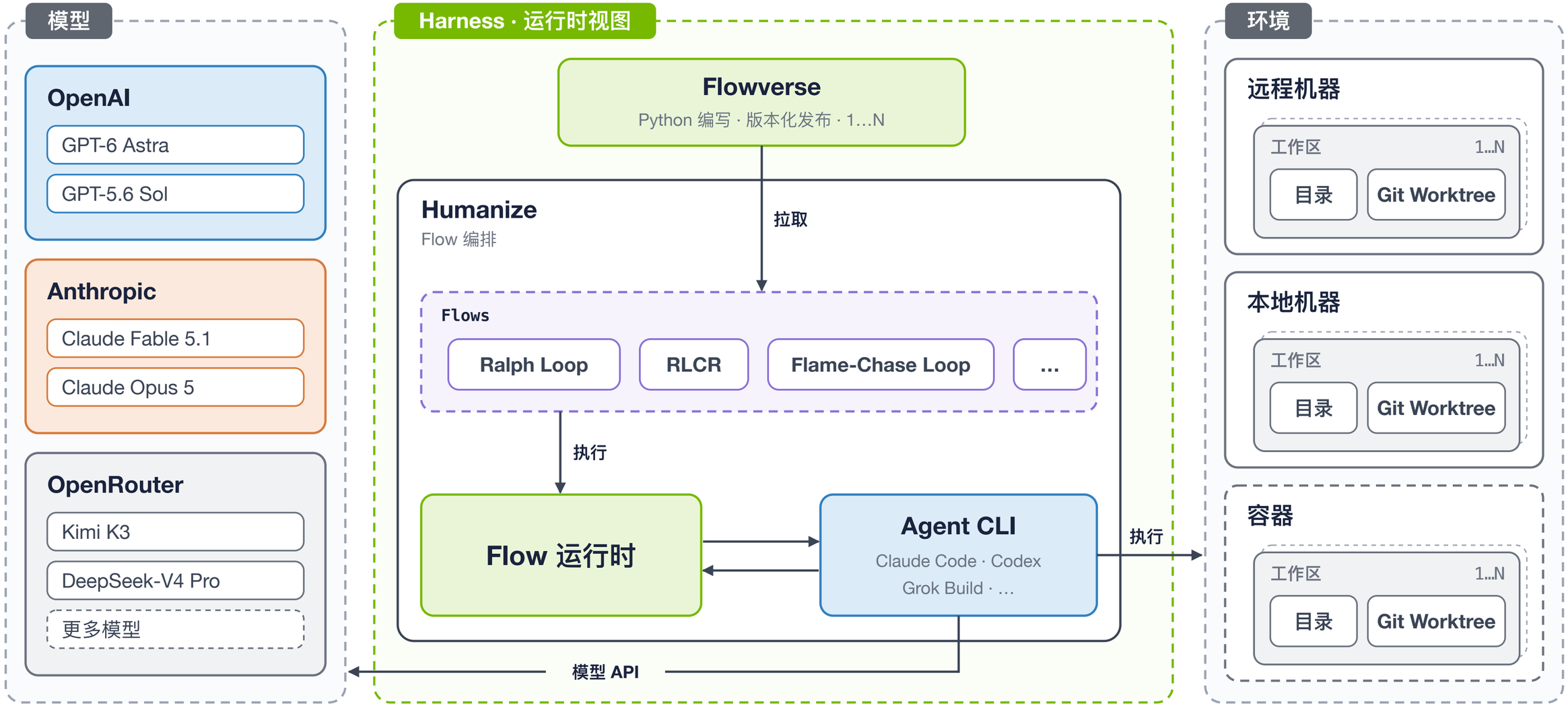
承载任意 Flow 的统一运行时，及由此催生的 Flow 科学

用原生异步 Python 编排任意 Flow · 统一驱动 13 款编程 Agent CLI · Flowverse 生态

github.com/humanfia/humanize (Apache-2.0) · github.com/humanfia/flowverse · docs.humanfia.ai/humanize

Humanize 2：Agent Flow 运行时

支持针对不同任务快速定制专属 Flow的系统



Humanize 2：Agent Flow 运行时

……并随附了极其丰富的开箱即用 Flow

内置 Flow

`ralph_loop`
每轮开启全新 Session 执行任务，仅靠代码库传递状态

`stateful_ralph`
在同一 Session 内每轮重发任务，保留会话历史

`continue_loop`
首轮发送任务后，在同 Session 内持续发送 continue 推进

`goal`
将任务一次性设为 Agent 的原生目标（/goal）

`flame_chase`
两个 Agent 在各自全新 Session 中轮流接力同一任务

`rlar`
执行者在单 Session 中工作，每轮由全新审查者检视成果

humanize1 · Humanize 1 的三阶段

`humanize1:gen-idea`
将初步想法拓展为结合代码库现状的详细草案

`humanize1:gen-plan`
编写与审查双方多轮打磨草案，收敛为一致的执行计划

`humanize1:rlcr`
在审查者把关下按计划构建代码，直至无可挑剔

元 Flow 与数学

`aot`
元 Flow：输入需求描述，输出经冒烟测试与审查的可用 Flow

`recursive_lean_prover`
递归规划、证明、对比、审查并归档 Lean 定理

flame_chasoid · Flame-Chase 变体

`agent_cleanup_ralph_loop`
每隔数轮由清理 Agent 提炼工作区状态的 Ralph Loop

`agent_cleanup_flame_chase`
定期由清理 Agent 提炼工作区状态的 Flame-Chase Loop

`fixed_interrupt`
两 Agent 以评测通过的实验为界交替接力，最后执行盲审

`parallel_git_pr`
三路独立克隆探索，仅当实测指标提升时才通过 PR 合入主干

`parallel`
三路报告驱动的交替 Agent 并行探索，仅第 1 路写入工作区

访问 github.com/humanfia/flowverse

快速上手

五个阶段，从安装命令到编写自己的 Flow

1

安装

```
uv tool install hmz
```

安装 hmz，并登录任一你常用的编程 Agent CLI。

2

首次运行

```
/flow
```

选好 Flow，为角色配好 Agent 与预算，输入任务。

3

无人值守

```
hmz exec -f …
```

将同样配置压成单行命令，用于脚本、cron 与 CI。

4

复用 Flow

```
Install more…
```

从 Flowverse 安装官方或社区 Flow（如 Humanize 1 的 RLCR）。

5

编写 Flow

```
@flow
```

用 @flow 编写异步 Python 函数：定义谁执行、谁审查、何时停止。

快速上手 · 安装

安装一条命令，复用你已登录的编程 Agent CLI

1 安装 | 2 首次运行 | 3 无人值守 | 4 复用 Flow | 5 编写 Flow

1. 通过 `uv tool install hmz` 安装命令行（或使用桌面端）。

   ```
   $ uv tool install hmz
   ```

2. 登录你常用的 Agent CLI；Humanize 复用本地登录态，不保存任何 API Key。

   ```
   $ claude auth login
   ```

3. 在临时 git 仓库中开始：Flow 中的 Agent 默认免审批执行。

terminal

```
$ uv tool install hmz
Installed 1 executable: hmz
$ npm i -g @anthropic-ai/claude-code
$ claude auth login
$ cd ~/tmp/humanize-demo && hmz
```

支持驱动以下 13 款编程 Agent CLI 中的任一款：

`claude` `codex` `cursor-agent` `agy` `grok` `kimi` `qwen` `opencode` `mimo` `mcode` `pi` `omp` `dsh`

NVIDIA

快速上手 · 首次运行

选定 Flow，配好 Agent 与预算，输入任务即刻启动

1. 在仓库中运行 `hmz`，默认进入 chat 模式。

```
$ hmz
```

2. 输入 `/flow`，搜索 ralph 并选择 ralph_loop。

```
› /flow
```

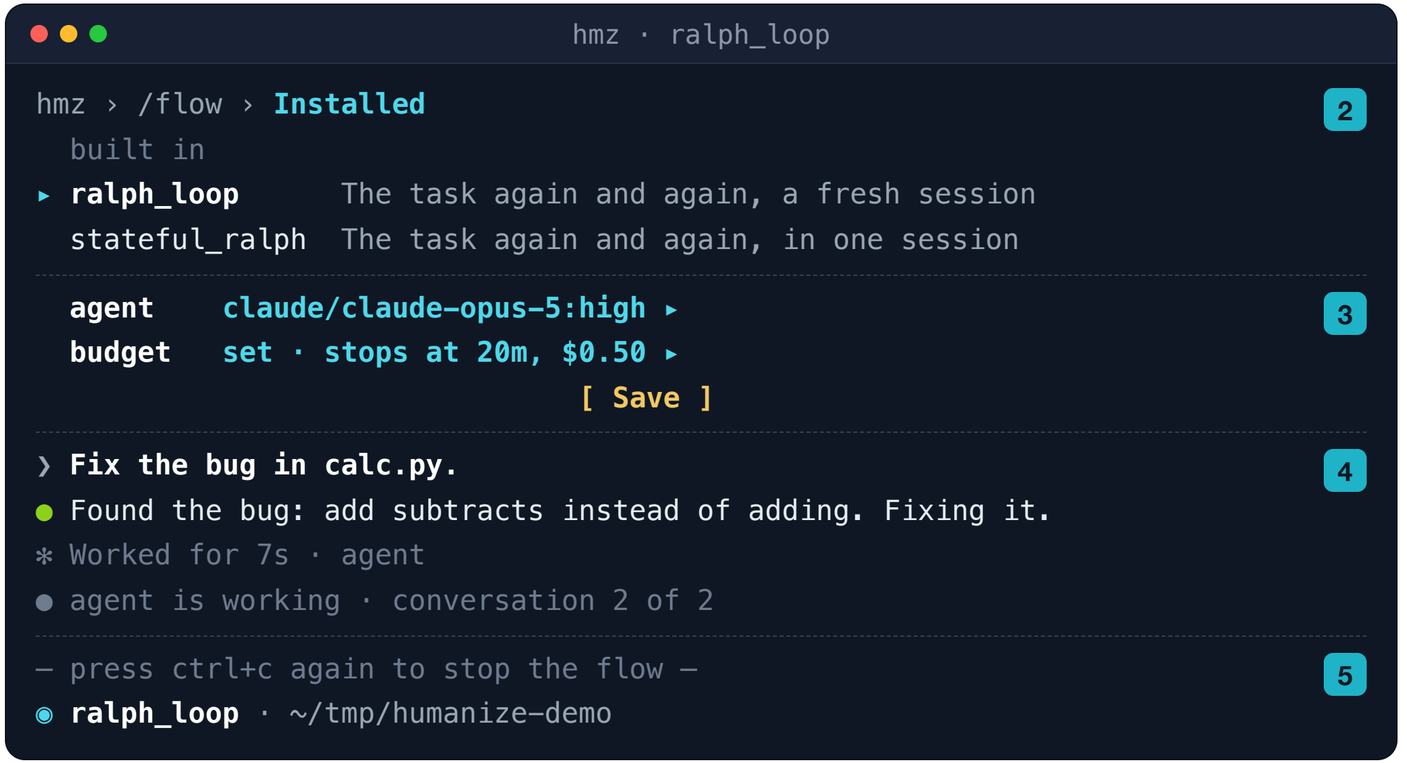
3. 为角色配置 CLI、账号、模型与推理强度，设定预算后点击 Save。

4. 输入任务并回车，Flow 随即启动。

```
❯ Fix the bug in calc.py.
```

5. 连按两次 ctrl+c 停止运行；用 `git diff start` 查看改动。

```
$ git diff start
```



快速上手 · 无人值守

将同样配置浓缩为单行命令，轻松接入脚本与 CI

✓ 安装 → ✓ 首次运行 → 3 无人值守 → 4 复用 Flow → 5 编写 Flow

```
$ hmz exec -f ralph_loop -a agent=claude/claude-opus-5:high -p budget.duration=2h,budget.cost=10 "$(cat TASK.md)"
```

1 指定要运行的 Flow

2 为角色绑定模型：`role=cli/model:effort`

3 设定时长或费用预算，先到即停

4 从文件读取任务描述

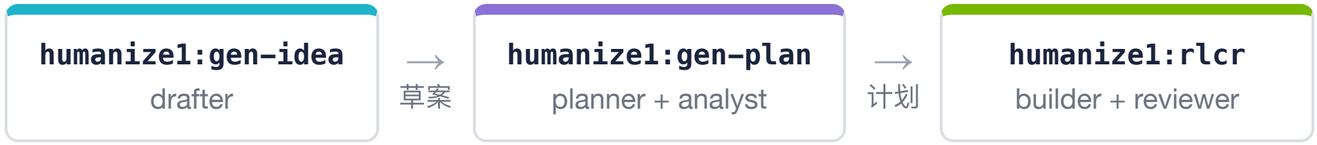
在 CI 中运行 · 每日构建作业

1 安装 hmz 与对应的 Agent CLI → 2 从 Secret 注入凭据登录 CLI → 3 执行同一条 hmz exec 命令 → 4 推送分支并自动创建 Pull Request

快速上手 · 复用 Flow

从 Flowverse 一键安装或直接拉起现成 Flow

1. 在 hmz 中选择 `/flow` → Install more… → official → humanize1 → Install。
2. 也可直接按 git ref 运行指定版本，无需提前安装。
3. 依次运行三阶段，阶段间通过可读写的文件传递草案与计划。



sh

```
2  hmz exec -f git+https://github.com/humanfia/humanize1-flow@v0.2.0#humanize1:gen-idea …

3  hmz exec -f humanize1:gen-idea -a drafter=claude/claude-opus-5:max "add undo/redo to the editor"
   hmz exec -f humanize1:gen-plan -a planner=claude/claude-opus-5:max -a analyst=codex/gpt-5.6-sol:max "add undo/redo to the editor"
   hmz exec -f humanize1:rlcr -a builder=claude/claude-opus-5:max -a reviewer=codex/gpt-5.6-sol:max "build the plan"
```

编织 Flow，从未如此简单

BASELINE PolyArch/Humanize

工作流深埋在繁杂的底层机制之中

118个文件

`commands/` ×5 `hooks/` ×14 `scripts/` ×22 `prompt-template/` ×64

`agents/` ×4 `skills/` ×6 `config/` ×2 `templates/`

setup-rlcr-loop.sh 1,541 行

parse_args "$@"
validate_model_and_effort
install_codex_hooks
…

loop-codex-stop-hook.sh 2,221 行

parse_state_file "$STATE_FILE"
CODEX_OUTPUT=$(codex exec ...)
parse_priority_findings
…

> **跨进程胶水代码**：Stop → shell → Codex CLI → JSON → jq，循环往复，直到所有门控全部放行

bash jq JSON 环境变量 进程信号 会话 ID 状态迁移 CLI 路由 模板加载 正则解析

OURS humanfia/humanize1-flow

代码即工作流，所见即所得

7个文件

humanize1/__init__.py 省略实现细节

```python
from hmz.flows import Agent, AgentCollection, flow

class Building(AgentCollection):
    builder: Builder
    reviewer: Agent

# Idea → grounded draft
@flow(agents=Drafting, envs=Where, params=Idea, name="gen-idea")
async def gen_idea(task, *, agents, envs, params, ctx):
    return await _idea(agents["drafter"], envs["workspace"], task, params)

# Draft → agreed plan
@flow(agents=Planning, envs=Where, params=Plan, name="gen-plan")
async def gen_plan(task, *, agents, envs, params, ctx):
    return await _plan(_Turns(agents, envs["workspace"], params), task, draft)

# Build ↔ review until done
@flow(agents=Building, envs=Where, params=Rlcr, name="rlcr", resumable=True)
async def rlcr(task, *, agents, envs, params, ctx):
    builder = agents["builder"]
    builder.on_permission_request(guard)
    session = await builder.spawn()
    while asking is not None:
        await _built(builder, session, env, asking)
        asking = await running.stopped()
```

快速上手 · 编写 Flow

一个 @flow 装饰器，加上原生异步 Python 函数

1. 在项目中创建目录：`.hmz/flows/twice/`。

```
$ mkdir -p .hmz/flows/twice
```

2. 用 `AgentCollection` 等声明角色与运行环境。

3. 在 `@flow` 下编写异步函数：用 spawn 开启会话，用 run 执行轮次。

4. 直接按名称调用，用法与其他 Flow 一致。

```
$ hmz exec -f twice -a builder=…
```

python

```python
# .hmz/flows/twice/__init__.py
from hmz.flows import (Agent, AgentCollection, EnvCollection,
                       FlowContext, FlowParams, LocalEnv, flow)

class Agents(AgentCollection):
    builder: Agent

class Envs(EnvCollection):
    workspace: LocalEnv

@flow(agents=Agents, envs=Envs, params=FlowParams)
async def twice(task: str, *, agents: Agents, envs: Envs,
                params: FlowParams, ctx: FlowContext) -> None:
    """Do the work, then read it back and fix what is wrong."""
    builder, workspace = agents["builder"], envs["workspace"]
    session = await builder.spawn()
    await builder.run(task, session=session, env=workspace)
    await builder.run(
        "Now review what you just did, and fix anything that is wrong.",
        session=session, env=workspace)
```


NVIDIA

当编织 Flow 足够简单：四大领域全面突破

基于 Humanize 2 构建专属工作流，在多项前沿任务中取得突破

01

数学与形式化证明

从奥数竞赛、定理基准到前沿猜想形式化

- IMO 2026 满分 (6/6)，PutnamBench 100%
- Lean-Eval 第一 (266/304)
- 首次形式化 4D Kakeya 等 8 道 OpenAI 难题
- 完成几何化猜想 Hamilton–Perelman 证明形式化

02

科学奥赛与学科推理

跨越信息、物理、化学、生物与量子计算

- IOI 2026 满分 (6/6)，IPhO 2026 全解 (23/23)
- IChO 2026 全形式化 (68/68)，IBO 2024 满分
- 量子算法与量子信息论两项基准均达 100%
- Physics Cup 80%，HLE 63%，SuperChem 62.4%

03

算子与系统性能优化

面向 GPU 底层算子的自动生成与深度调优

- MLSys FlashInfer 全赛道超越人类 SOTA
- SOLExecBench L1 第一，产出 71 个榜首算子
- MSA Prefill 最高加速 14×，VAE 解码 2.7×
- 优化 40+ SGLang 算子，修复 cuDNN 隐蔽 Bug

04

机器学习工程

多 Agent 轮流接力突破端到端建模上限

- MLE-bench 夺牌率 78.2%（五对比方案第一）
- 金牌率 53.8%，90.7% 任务超人类中位数
- 6 种模型组合较单 Agent 均升 6.4 个百分点
- 前身在 39 场 Kaggle 实赛 3 次进最终榜前 5%

相关技术报告与论文

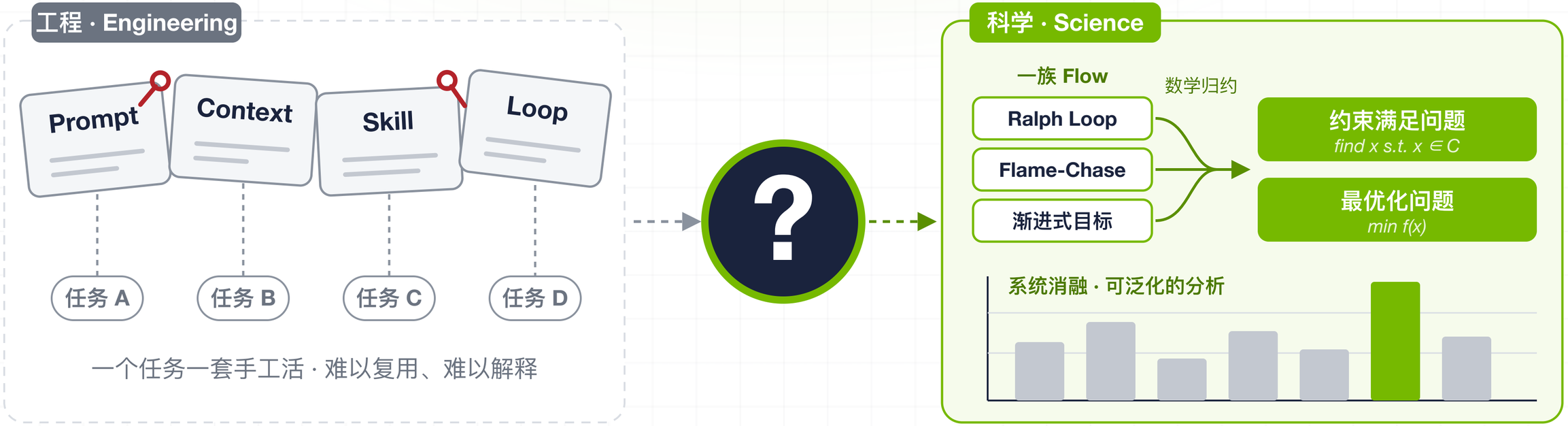
- Humanize 判别工程 — arXiv:2610.08900
- HOA 奥赛多会话工作流 — arXiv:2610.13632
- HMA 多 Agent 会话交替 — arXiv:2610.13494
- 几何化庞加莱猜想 Lean 形式化 — arXiv:2610.13897
- Parason 推理并行性剖析 — arXiv:2608.24658
- Humanize-Prover 可扩展数学工程 — in submission

其中奥赛、算子与机器学习成果均由约 10 人小团队于今年（6–10 月）四个月内完成

NVIDIA

Flow：是工程还是科学?

仅仅是又一种 Prompt / Context / Skill / Loop 工程,
还是超越它们的新范式?



从 Flow 工程到 Flow 科学

以问题数学归约为基础，开展可泛化的系统级消融与分析

XXX 工程

一个 XXX 绑定一个任务，充满繁琐的脏活累活

→

Flow 工程

针对单一特定任务定制专用 Flow

→

Flow 科学

一族 Flow（如 Ralph Loop、Flame-Chase Loop、渐进式目标）可泛化至能**归约为同一数学模型**的所有任务（如约束满足问题、最优化问题）

……并且我们能在此基础上开展**系统的消融实验与理论分析**。例如对 Flame-Chase Loop 消融：

1 参与模型 **2 模型顺序** **3 调用比例** **4 并行模式** **5 协作模式** **6 任务追踪** ……

Flow 科学 = Agent 的管理艺术 *The art of agents' management*



同样的 token，不同的人 / 不同的使用方式，**最后的产出差很多**

同样的人才，不同的组织架构 / 管理方式，**公司最后的成就也不同**

核心发现

不同问题，匹配不同 Flow

软件工程实现问题

→ **RLCR**

在审查者把关下按计划构建代码，直至无可挑剔

优化问题

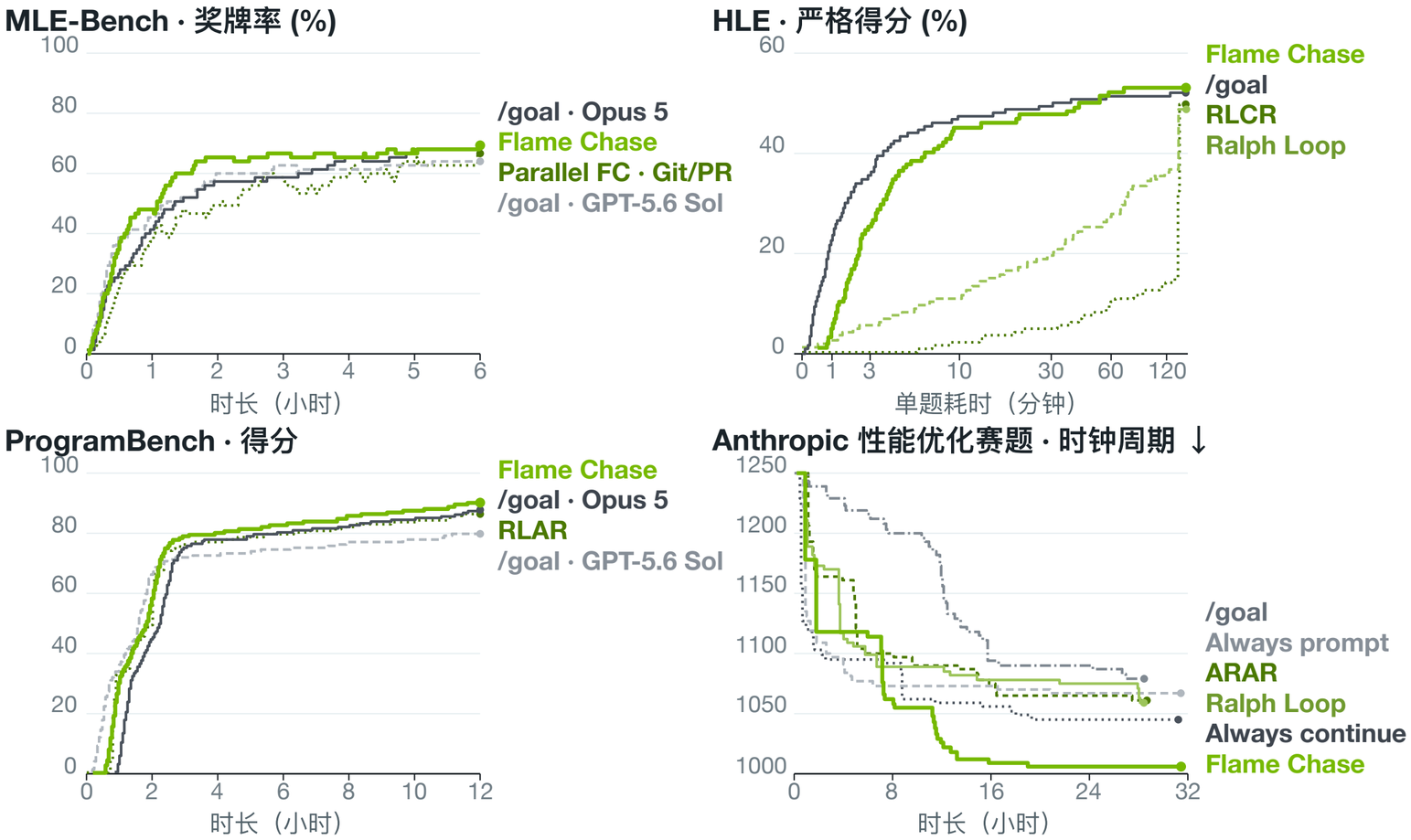
→ **Flame-Chase Loop**

两个 Agent 在各自全新 Session 中轮流接力同一任务

更多问题类型呢?

判定问题？ 预测问题？ 博弈问题？

每一类问题，都需要匹配自己的 Flow



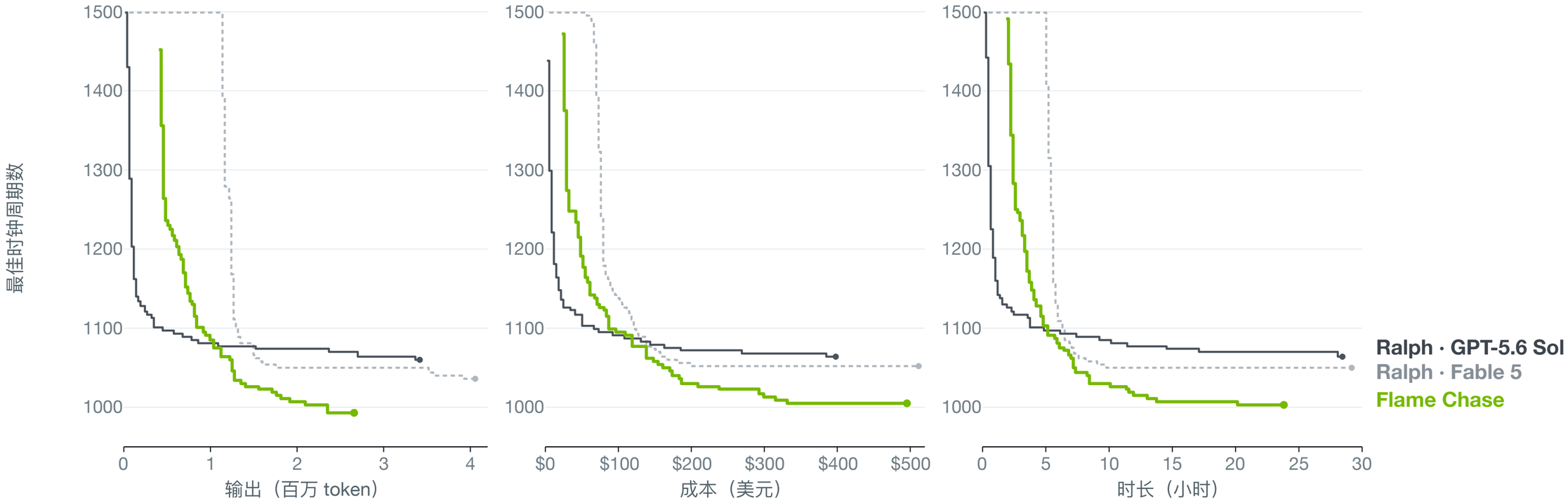
Humanize Flow（最亮 = Flame Chase）

/goal 等基线

NVIDIA

核心发现

多样性越高，得分越高



实验设置

- 任务：Anthropic 原版性能优化赛题
- 指标：最佳时钟周期数，**越低越好**
- 横轴：输出 token、成本、时长

对比的 Flow

Ralph · GPT-5.6 Sol	单一模型，每轮全新 Session
Ralph · Fable 5	同一 Ralph Loop，换成 Fable 5
Flame Chase	两个 Agent 轮流接力同一任务，只交接代码库

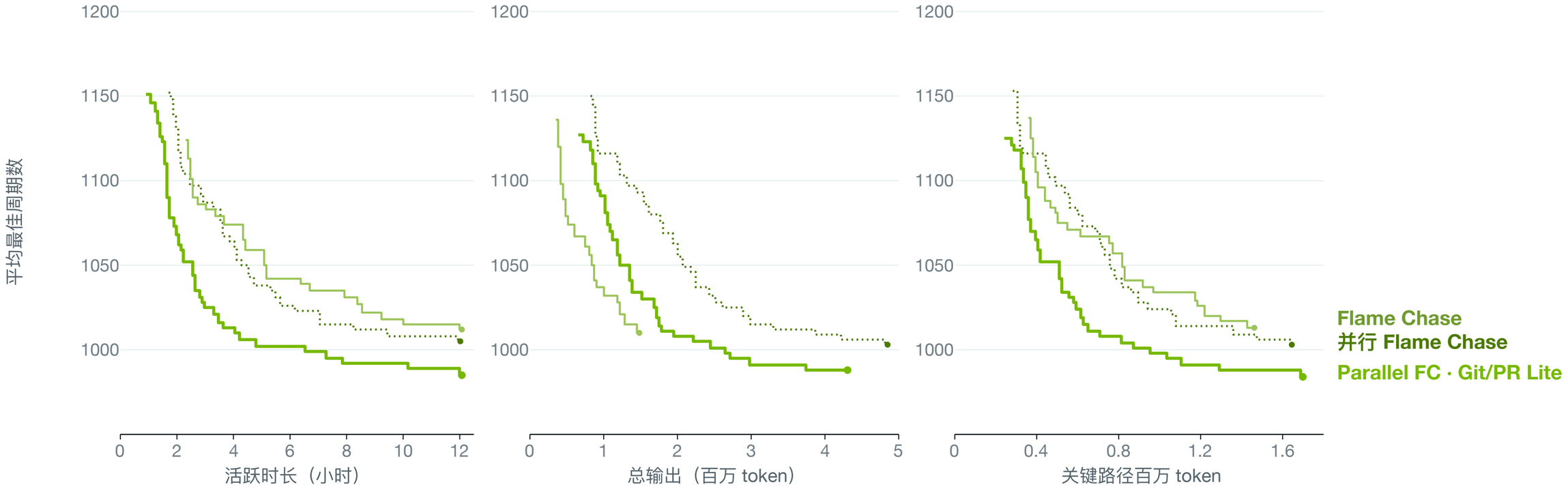
结论

- 单模型 Ralph 停在 **≈1,040–1,060** 周期
- Flame Chase 降到 **≈1,000**，输出更少（≈2.7M vs 3.4–4.1M token），用时更短（≈24h vs 28–29h）
- **不同模型失误模式不同**：轮流接手，盲区不会一路累积

数据：Anthropic 原版性能优化赛题（github.com/anthropics/original_performance_takehome）· 数值为图中读数 · Flow 定义：github.com/humanfia/flowverse

核心发现

协作越高效，性能越出色



实验设置

- 任务：Anthropic 原版性能优化赛题
- 指标：平均最佳时钟周期数，**越低越好**
- 横轴：活跃时长、总输出 token、关键路径 token

对比的 Flow

Flame Chase	单路：两个 Agent 轮流接力
并行 Flame Chase	三路并行，靠报告互通，仅第 1 路写入工作区
Parallel FC · Git/PR Lite	三路独立克隆，实测指标提升才经 PR 合入主干

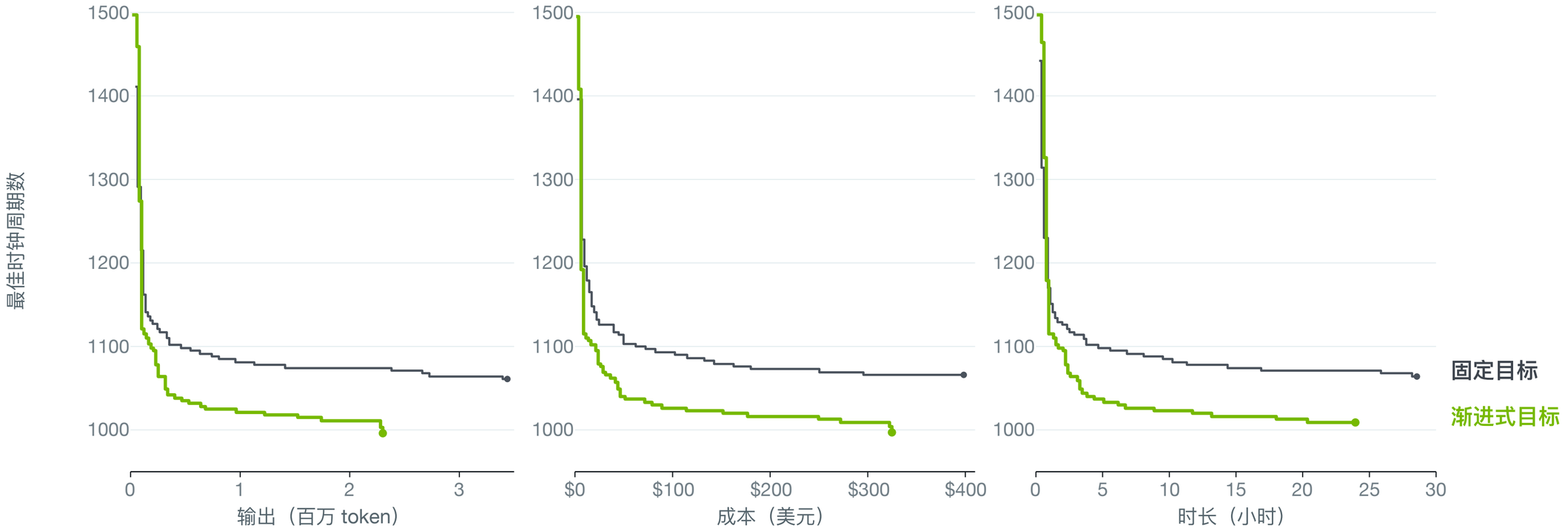
结论

- 12 小时内 Git/PR Lite 降到 **≈985**；Flame Chase ≈1,012，并行 Flame Chase ≈1,005
- **光并行不够**：报告式并行多花 3 倍以上输出 token（≈4.9M vs 1.5M），收益有限
- **以实测指标把关合入**，多路探索才能真正叠加

数据：Anthropic 原版性能优化赛题 · 数值为图中读数 · Flow 定义：flame_chasoid:parallel / parallel_git_pr，github.com/humanfia/flowverse

核心发现

渐进式目标（Progressive Goal）能防止 Agent 迷失方向



实验设置

- 任务：Anthropic 原版性能优化赛题
- 指标：最佳时钟周期数，**越低越好**
- 横轴：输出 token、成本、时长

对比的目标设定

固定目标 整个运行始终面对同一个目标

渐进式目标 目标随进展逐步推进

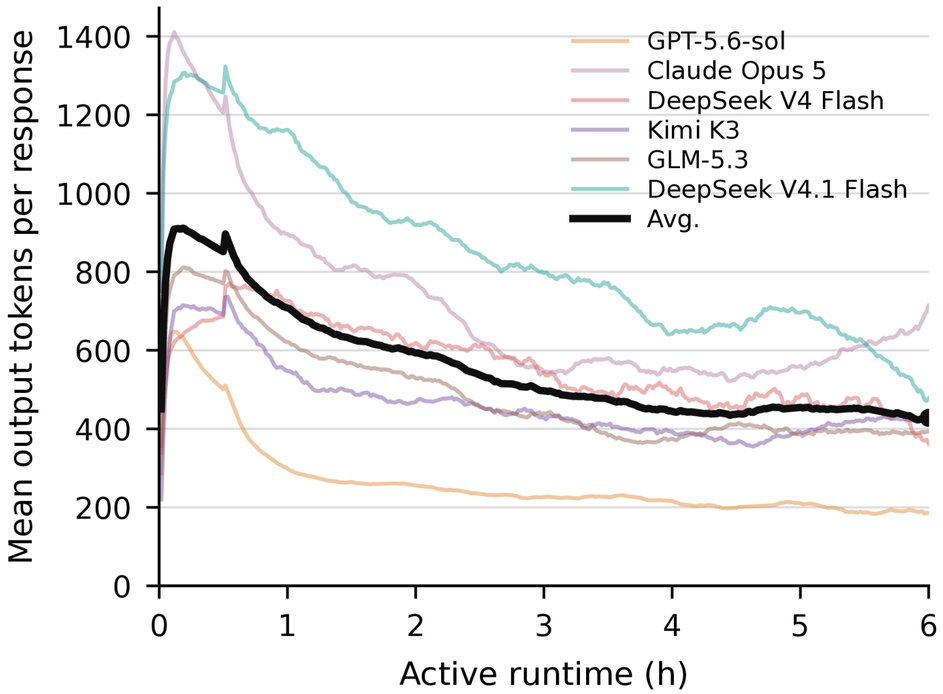
结论

- 固定目标停滞在 **≈1,060** 周期；渐进式目标降到 **≈1,000**
- 还更省：输出 ≈2.3M vs 3.45M token，成本 ≈$325 vs $400，时长 ≈24h vs 28.7h
- **阶段性目标**让 Agent 在长程运行中保持方向

数据：Anthropic 原版性能优化赛题 · 数值为图中读数

为什么不能只靠 /goal？思考预算在缩水！

长程减速：单个会话越跑越久，Agent 每次回复的输出 token 越来越少，收益随之停滞



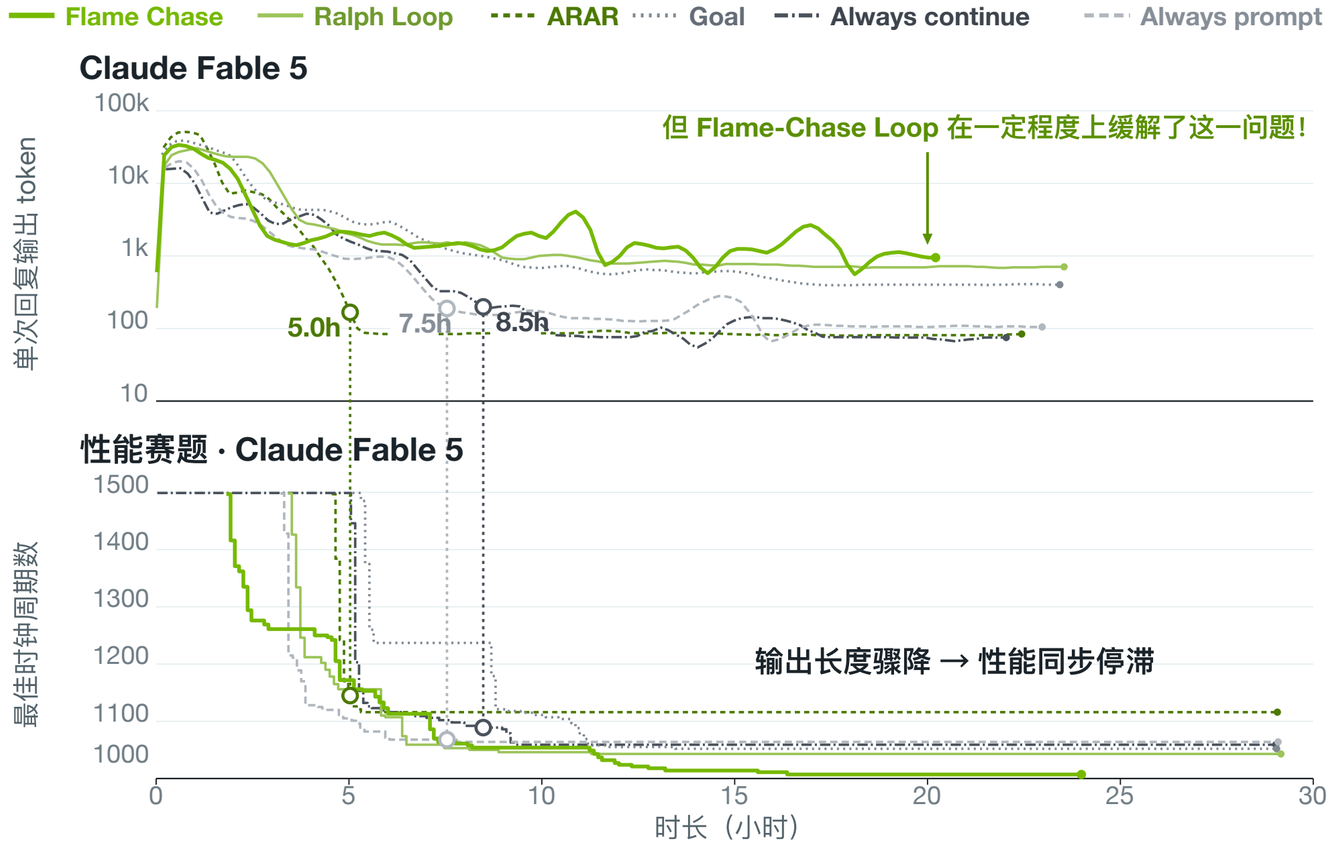
单个 /goal 会话：单次回复平均输出 token（30 分钟滑动窗口）

–29.8~45%

6 个模型，0–2h → 3–5h 单次回复 token 降幅

86.4% vs 60.7%

重置上下文（HMA）vs 单会话 /goal 保留的单次回复 token



别拉长一个会话：**结束它、保留成果、重新开始**

左：HMA (arXiv 2610.13494) Sec. 1–2, Fig. 2–4, App. D.1 Table 6, App. E（降幅 29.8%–45.0%）· 思考预算以单次回复平均输出 token 衡量 · 右：Claude Fable 5 长程实验

PART 3 · 第三步

Humanize NEXT：可扩展 Flow 基础设施

从双 Agent 轮流接力，到大规模动态 DAG 协同并发

Exomyth 全局可视化 · CogAnchor 透明远程执行 · 动态 DAG 并发调度

案例研究：Humanize-Prover 可扩展数学工程（in submission）· 几何化庞加莱猜想 Lean 形式化 · arXiv:2610.13897

从 Flow 科学到系统级需求

实验规律对底层基础设施提出了哪些核心要求

Flow 科学的实验发现		基础设施必须提供的能力	Humanize 现已支持
重置上下文优于超长单次会话 Ralph Loop 与 Flame Chase	→	Session 应当轻量、用完即弃；状态沉淀在代码工作区中	spawn / run；worktree 与 reflink、基于 git ref 的快照与回滚
模型多样性带来显著收益 Fable 5 与 GPT-5.6 交替优于单模型	→	统一接口屏蔽异构模型池，支持故障自动回退	接入多款 CLI 与 LiteLLM；支持跨 CLI · 账号 · 模型的回退链
AND / OR 双重并行行之有效 多分支并行探索与动态子任务拆解	→	支持投机执行、暂停与取消的动态 DAG 调度器	asyncio 调用树（gather, TaskGroup）；嵌套预算；上层构建 DAG 控制器
验证环节成为系统瓶颈 环境与检查器吞吐须跟上 Agent 并发	→	运行环境与验证器需随 Agent 同步横向扩展	本地 / SSH / Docker / Swarm / Slurm；按角色限制 CPU · GPU · 内存配额

核心技术：Exomyth（神话之外）

全局可视化引擎



在同一时间轴上，从顶层 Flow 调用一路下钻到每轮 Agent 会话及其拉起的系统进程。

单次运行的终端时间轴与火焰图：支持缩放、平移与按名搜索，可流畅渲染 50,000 个 span。

核心技术：CogAnchor（识刻锚）

透明远程执行



Agent 平面

Agent CLI 调度器

驱动各 CLI

Agent CLI

- Claude Code
- Codex
- Grok Build

Hook 系统

标准 · 权限

Provider · 配置 · 数据

技能 · 工具 · 子 Agent

CogAnchor Hook

动作

反馈

Humanize 控制面

CogAnchor

动作路由网关

1 **接收动作**
来自选定的 Agent CLI

2 **映射目标**
环境 + 工作区

3 **返回反馈**
回传至 Agent CLI

目标 · 健康状态

环境管理器

管理 · 监控 · 启停本地容器

生命周期 · 健康状态 · 启停

执行环境

远程机器

CogAnchor Worker

工作区 1…N

目录 | Git Worktree

本地机器

宿主机工作区

目录

Git Worktree

容器

CogAnchor Worker

工作区 1…N

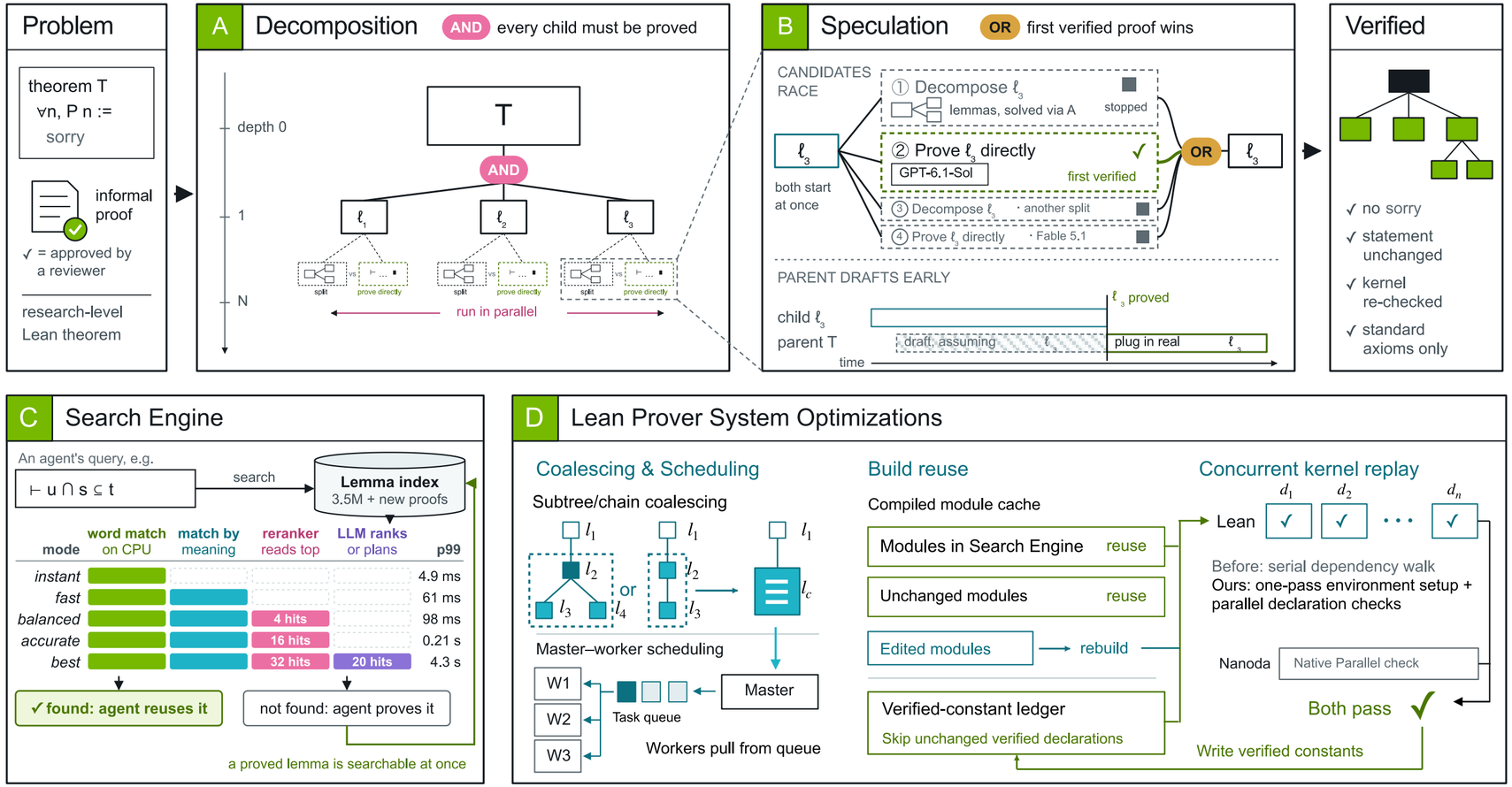
目录

Git Worktree

NVIDIA

案例研究：Humanize-Prover

形式化证明即并发调度



系统总览：**(A)** 引理 AND 拆解　**(B)** 多路 OR 投机　**(C)** 实时引理检索引擎　**(D)** Lean 核心和基建优化

Humanize-Prover

四维挂谷猜想（OpenAI 问题 074）首次 Lean 形式化

141
峰值并发 Agent

40.6 小时
墙钟时间

296
个引理，**241** 个通过

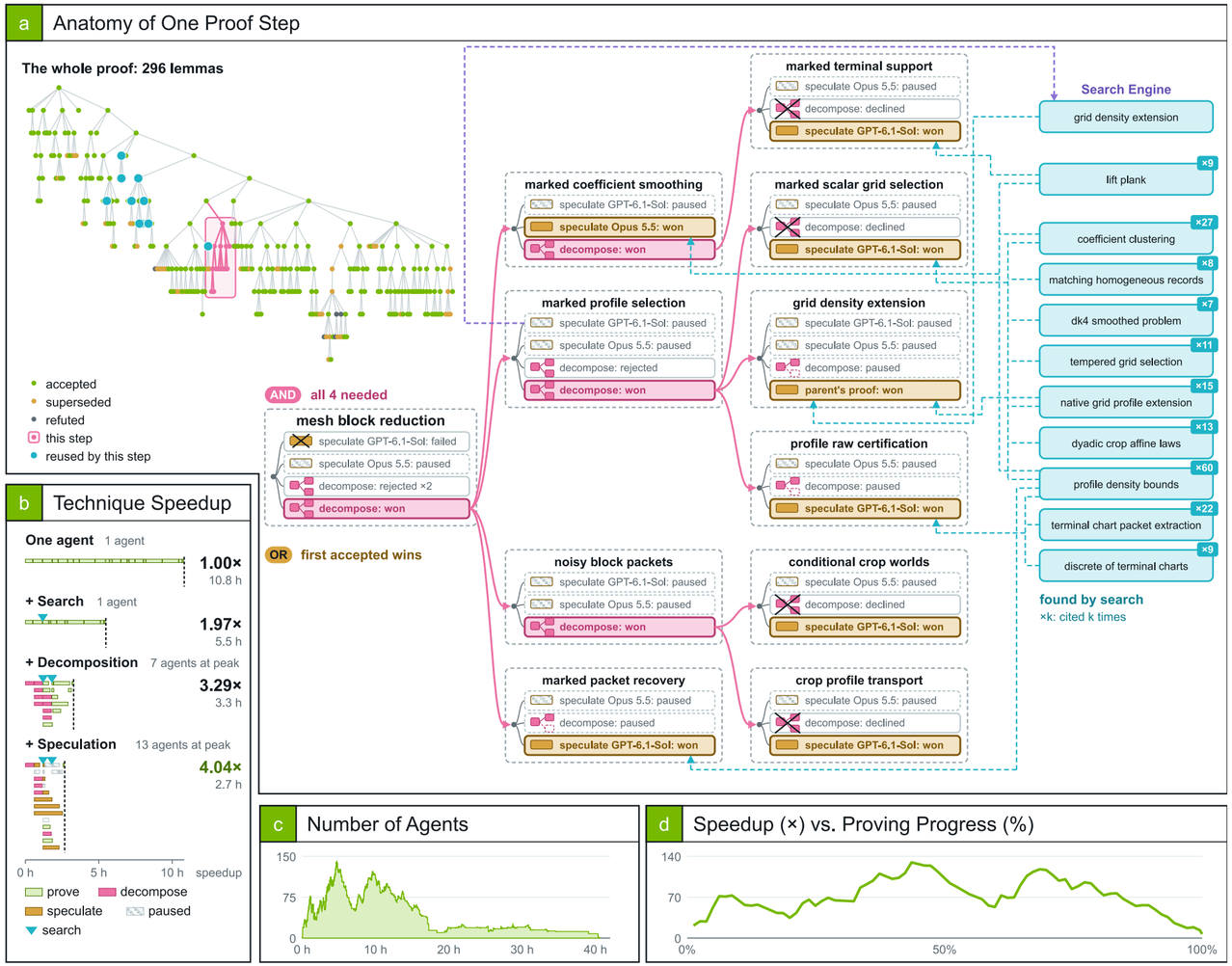
138 万行
Lean 证明

如果串行来做，要多久?

按每行 Lean 约 1,000 token（含探索）：138 万行 ≈ 13.8 亿 token

人类专家 每分钟 200 token，不停	**≈ 13 年**
单个 Agent（Opus 5.5） 每秒 100 token，不停	**≈ 160 天**
Humanize-Prover 动态 DAG，峰值 141 并发	**40.6 小时**

Speedup：比单 Agent 快 **≈ 94×**



Humanize-Prover（in submission）Sec. 3.6 · 按每行 Lean 约 1,000 token：138 万行 ≈ 13.8 亿 token · 人类 200 token/分钟 ≈ 11.5 万小时 · 单 Agent 100 token/秒 ≈ 3,833 小时

现在 → 未来

从单机运行时迈向百万级 Agent 基础设施

当前已实现 hmz 现已具备的能力

执行引擎	异步调用树 + 只追加日志；支持断点续跑
运行环境	local · SSH · Docker · Swarm · Slurm · Apple container；按角色限制 CPU/GPU/内存，回收残留容器
安全沙箱	Fence：Landlock + seccomp / Seatbelt，统一网络代理
远程执行	CogAnchor：系统调用按需镜像、NAT 打洞、EasyTier 组网
预算与容错	嵌套时间/费用/Token 预算；跨层回退链

当前规模：通过上层 DAG 控制器支撑高达 **194** 个并发 Session

→

下一步演进方向 基础设施的未来规划

核心原语	将动态 DAG 与投机执行沉淀为原生一等原语
集群调度	配额感知的跨机全局调度
运行环境	热环境池复用；跨运行结果缓存
控制平面	多机分布式日志与协同控制面

目标：**百万级** Agent 基础设施

NVIDIA

加入社区：与我们共建 Humanize 生态

从核心运行时、跨网互联到千行百业的领域专属 Flow，期待你的 PR

01

UI/UX 与移动端体验

- 打磨更直观友好的交互体验与状态可视化
- 构建适配移动端的轻量级监控与操控界面

前端 / 交互设计

02

跨网络客户端–服务端互联

- 集成 Tailscale、EasyTier 等主流异地组网工具
- 实现随时随地跨网安全连接远端 Agent 运行时

网络 / 系统工程

03

探索更多特定负载 Flow

- 面向 3D 建模、视频与动漫制作的多模态工作流
- 面向大规模软件跨语言移植等长程工程的专属 Flow

Agent 创作者 / 领域专家

如何开始贡献

1
动手前先在 Issue 区认领任务，
避免重复开发

→

2
较大改动请先开 Issue
与维护者对齐技术方案

→

3
提交代码 PR 请遵循
Conventional Commit 规范

→

4
自研 Flow 托管个人仓库，
提 PR 收录至 Flowverse

核心仓库 github.com/humanfia/humanize (Apache-2.0)　社区索引 github.com/humanfia/flowverse　文档 docs.humanfia.ai/humanize

深入探索 Humanize

Humanize 的宏大愿景

Humanize 的愿景：1 个统一的 Flow 框架

100T	1K	1M	1M	1M
参数模型	TPS 吞吐	并发 Agent	运行环境与容器	分钟持续运行

合力攻克一个终极难题　Ad Astra！让我们一起 Scale！

Humanfia 团队 · humanfia.ai/about


Sihao Liu


Ligeng Zhu


Zijian Zhang


Dongyun Zou


Changye Li


Dong Zhou

Zheng Du


Yixin Dong


Junxian Guo


Song Bian


Xiaoyu Zhang

Yahui Cui

Zhengyang Zhang


Jing Xiong


Hongzhou Lin

Jui-Hui Chung


Mark Kim


Menghan Li


Yitong Liu

NVIDIA.