ICML 2026

JIT Compiling Computer-Use Agents

Latency-optimizing planning and scheduling by compiling tasks to code

Caleb Winston

Ron Yifeng Wang

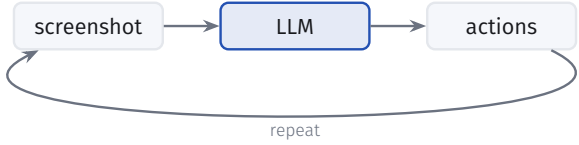
Azalia Mirhoseini

Christos Kozyrakis

THE PROBLEM

Computer-use agents are stuck in a static loop

Prior art agents typically run a static while loop program:



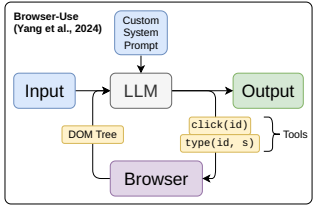
- **LLM inference at every step** of the critical path
- Wrong tool ordering dominates failures
- Forced sequential, even on parallel work

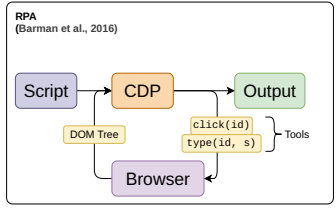
73% of Browser-Use latency inside LLM calls

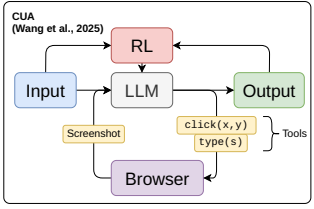
80% plan failures without structural guarantees

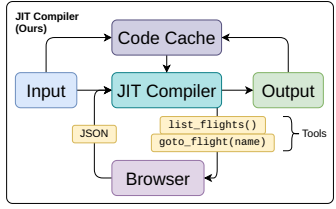
THE HYPOTHESIS

A cost-optimal program can be compiled for every task.





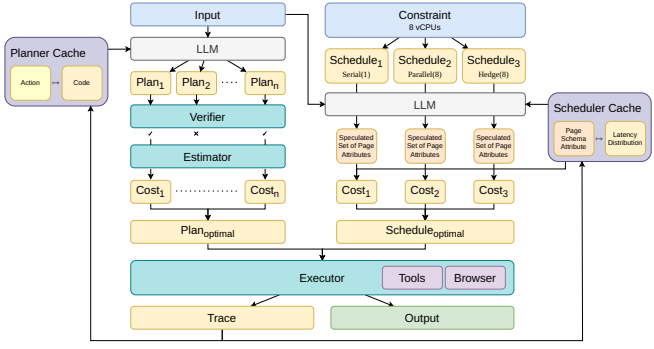




We translate the task directly into code over cached, reusable tools (`list_items`, `add_to_cart`) with a cost function to optimize code performance.

SYSTEM OVERVIEW

A JIT compiler for agents, in three parts



1 PROTOCOL

pre/postcondition invariants make tool use statically checkable

2 PLANNER

generate code candidates, verify, pick the minimum-cost plan

3 SCHEDULER

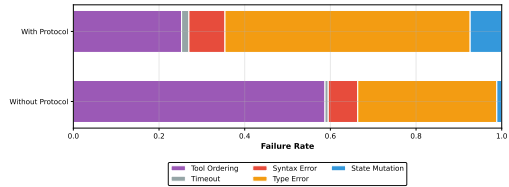
choose serial / parallel / hedge by expected latency

Tools that carry their own contracts

Each tool declares the state it requires and the state it guarantees:

```
pre   : { page: "*" }
post  : { page: "detail",
          selectedItem: $id }
```

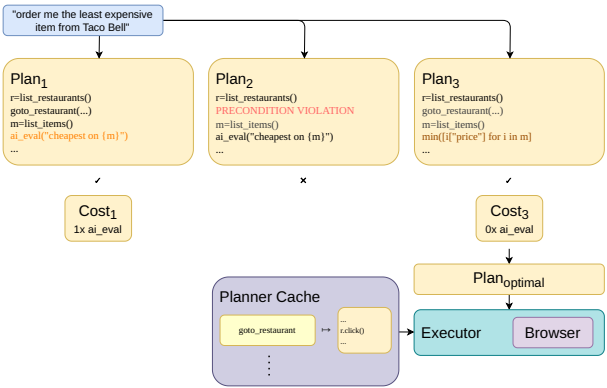
A sequence type-checks when $\textbf{post}_i \subseteq \textbf{pre}_{i+1}$.
Bad tool orderings are rejected without requiring further inference for chain-of-thought.



Total failure rate **80%** → **43%**

Tool-ordering errors **59%** → **25%**.

Some plans are correct. Few are fast.

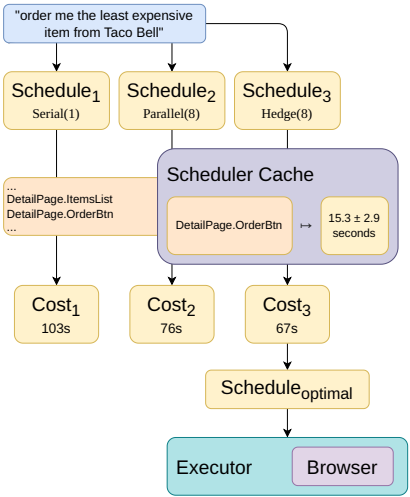


- Sample *k* candidates in parallel
- Walk each CFG: verify state flow, estimate cost
- Penalize LLM calls in loops; keep the **minimum-cost** plan

5.3× gap between best and worst valid plan

vs. Browser-Use:
10.4× faster, +28% accuracy.

The fastest schedule is not the obvious one



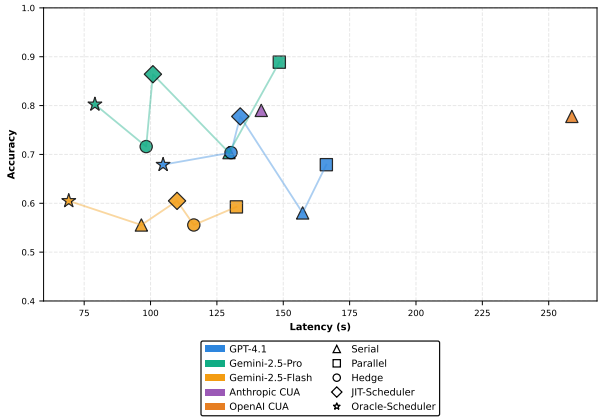
- Three strategies: **serial**, **parallel**, **hedge** (first-of-*k*)
- Predict which page elements the plan touches
- Monte Carlo over **learned latency distributions**; pick the lowest expected latency

vs. OpenAI CUA: 2.6× faster, +9% accuracy.

Beats the latency-greedy oracle by 6–10% accuracy.

RESULTS

Adaptive selection wins the frontier



Across 37 tasks and 5 applications, JIT-Scheduler sits on the upper-left frontier:

- Higher accuracy than *any* fixed strategy
- Lower latency than both shipped CUAs
- Training-free

TAKEAWAY

Treat the agent as a program to cost-optimize

PROTOCOL

Contracts on tools turn correctness into a static check.

PLANNER

Picking the cheap plan is worth up to 5.3×.

SCHEDULER

Picking the right schedule beats fine-tuned CUAs.

Compile computer-use tasks into cost-optimal programs:
Pareto-better latency and reliability.

Caleb Winston
calebwin@stanford.edu
caleb@arker.ai