

AGENTICOS 2026 · PRAGUE

AgentPProf: Semantic Profiler for AI Agents

Yusheng Zheng · Chaokun Chang · Yu Mao · Tianyuan Wu · Yuxi Huang

Tao Ma · Wenan Mao · Shuyi Cheng · Andi Quinn · Wei Wang

github.com/eunomia-bpf/agentsight

build deployment system

deploy branches

diagnose authentication

validate deployment

...

...

Developers need answers across many agent runs

hours or days

per agent run

weeks or months

of trajectories per team

QUESTIONS SPAN MANY TRAJECTORIES

Which tasks consume the most token budget?

Where do failures concentrate across workflows?

Which activities trigger unsafe system effects?

Reading every transcript, or running an LLM judge on each, gets costly at scale.

Agent observability needs profiling, not only debugging

WHAT AGENT TOOLS GIVE TODAY

Tracing

LangSmith, Langfuse,
Phoenix

A span tree per run: *what happened in this execution.*

Input analytics

Datadog LLM Observability
Input distribution: *“30% of users ask code questions.”*

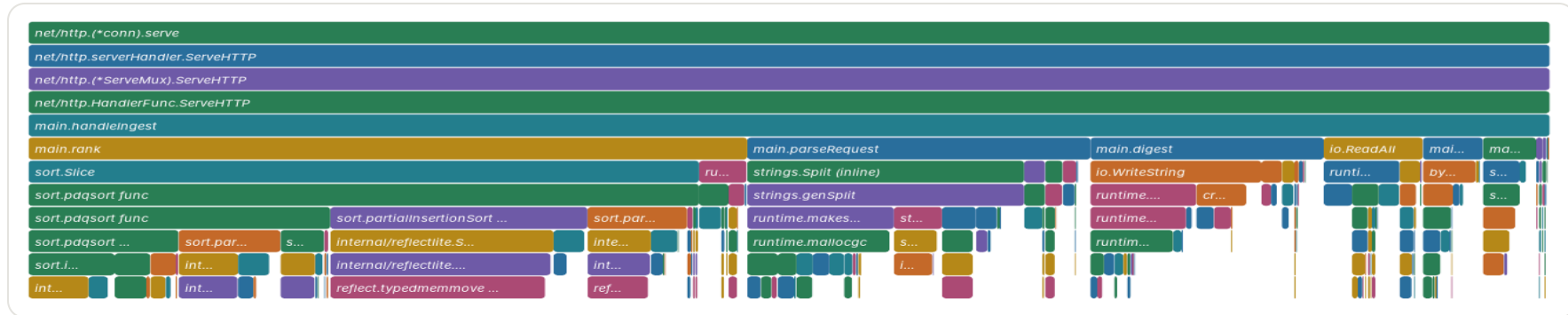
Missing: resource attribution

“Review tasks consume 40% of the token budget.”

Request-level tags do not reach the tool calls and system effects below them.

We want a **profiler** like system profilers, for AI agents: aggregate resource use across runs and attribute it to the responsible work.

Example: a system code profiler



Paper Figure 1a · CPU time in a Go HTTP service, focused on `main.handleIngest`, drawn by the same renderer as the agent profile on the next slide.

① Width = total cost

Here CPU time: `main.rank` is the widest child.

② Parent contains children

`sort.Slice`'s time also counts for `main.rank` and `handleIngest`.

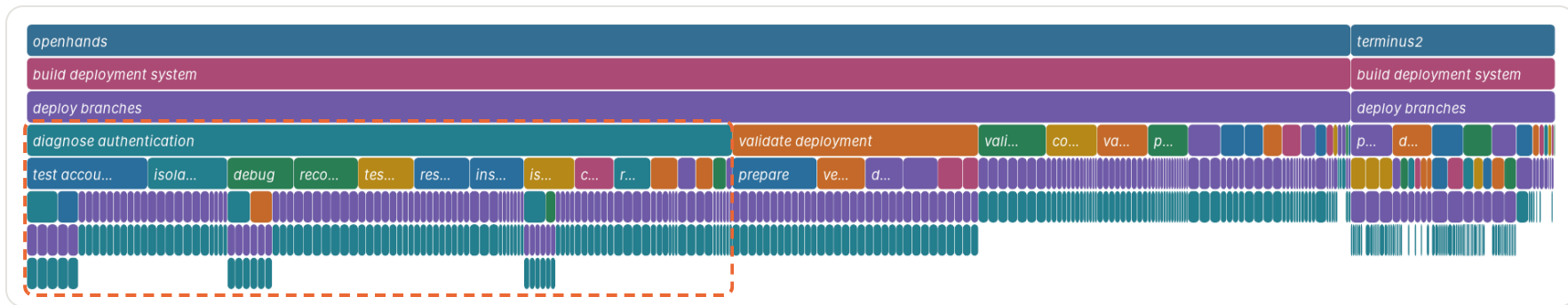
③ Equal stacks fold

Samples from different moments with the same stack merge into one box.

The frames are free: every CPU sample already carries its call stack.

Example: an AI agent profiler, on three failed runs

One Git-deployment task, three agent runs, all failed: 4.6M tokens in total.



Paper Figure 1c, built by AgentPProf · three runs folded, width = tokens. Dashed: *diagnose authentication* and everything under it.

46%

of the three runs' tokens go to
diagnose authentication

What happened

In all three runs, agents retry
SSH variants instead of fixing
the actual credential problem.

How to fix

Validate credentials early, or
bound retry depth, in *diagnose
authentication*.

How can we create this from raw agent logs?

Agent trajectories lack stable names and a call-stack hierarchy

C1 The responsible entity is task intent,
not a code path

Prompts are natural language: the three runs express *diagnose authentication* through different prompts and commands. No common string to fold on.

C2 No runtime hierarchy for attribution

A sequence of prompts may be one task or several; a task may be part of a larger one. Nothing marks the boundaries.

R1 Cross-layer projection

connect system effects
(processes, files, network) to
intent-level categories

R2 Stable identifiers

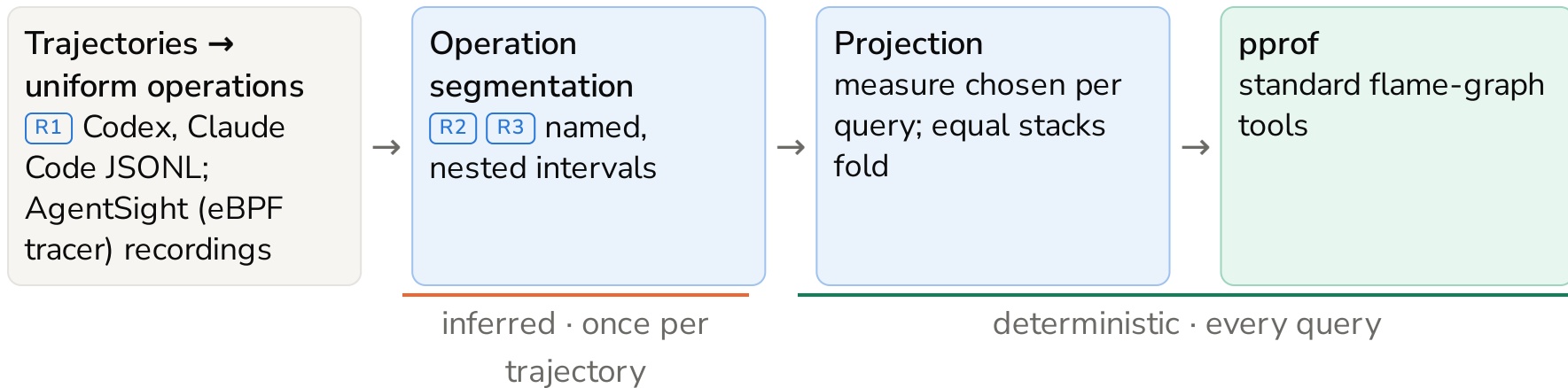
short, repeatable names
derived from prompts

R3 Hierarchical attribution

nesting built from data, not
execution

AgentPProf builds the missing stack from the trajectory

The semantic operation stack model: uniform operations replace samples, and operation stacks replace the call stack.



Implementation. A Rust CLI (~9.8K LOC) reads agent histories and AgentSight eBPF recordings as they are written, so profiles can be built while the agent runs, and emits a standard pprof profile for `go tool pprof`.

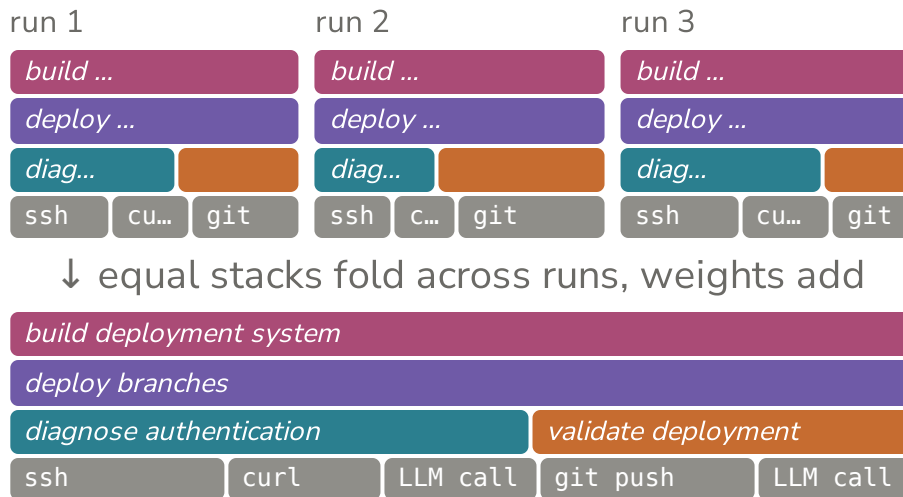
Every activity is an operation; tasks form its stack

OPERATIONS FROM ONE RUN (SCHEMATIC)

kind	content	measure
prompt	<i>"add password auth"</i>	—
LLM call	model reply	tokens
tool call	<code>curl -u dev</code> <code>.../repo</code>	duration
system effect	<code>curl</code> process · network	events

One schema: string fields and additive measures.
Activities differ only in which fields carry values.

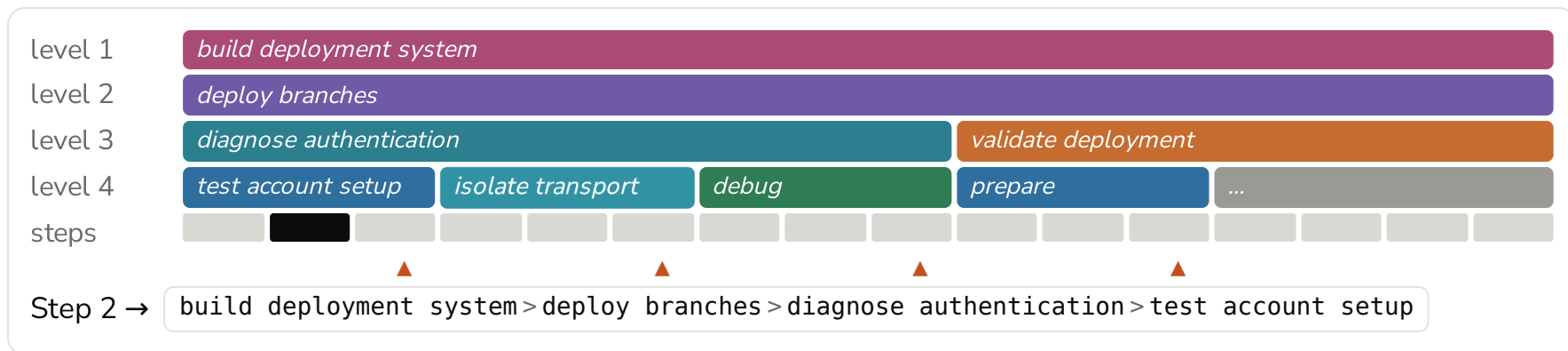
OPERATION STACKS: TASK PATH, CALL AS THE LEAF



But trajectories record neither the task names nor their nesting.

Find where responsibility changes, name it, recurse

A task occupies a contiguous span and decomposes into contiguous subtasks, so only the transitions need to be found.



Schematic; frame names from Figure 1c. ▲ = transition point.

- 1 Pick zero or more transition points ▲ → consecutive children
- 2 Name each child with a short verb phrase, normalized so the same work gets the same name in every run R2
- 3 Recurse into each child R3; pick none → the branch stops

Any segmenter fits: an LLM agent (Codex in the paper), a small model, or a statistical rule.

Equal names fold across runs: this is where *diagnose authentication* in the motivating profile comes from.

The profile points at real problems, and at the fix

All annotations, outcomes and scores stay hidden from every method until its output is produced.

CAN A PROFILE FINDING GUIDE A REAL REPAIR?

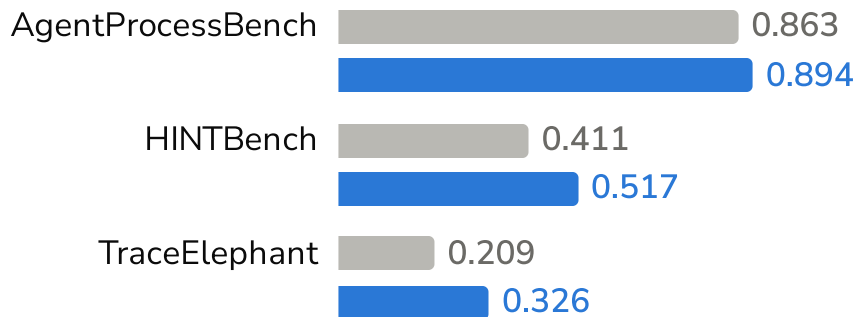
1 · **Profile** 8 ToolSandbox (tool-use benchmark) runs: one recurring failure stands out, invalid tool-call IDs (5 of 21 tool calls).

2 · **Locate** An analyst agent that sees only the pprof output, no raw traces, picks the target: the converter that turns tool calls into Python.

3 · **Fix** One line removes all these failures; on 23 held-out scenarios, agent tokens drop **19%** without degrading task quality.

DOES THE PROFILE HELP FIND THE FAULTY STEP?

MAP (0–1): how high the true faulty step ranks; 1 = always first.



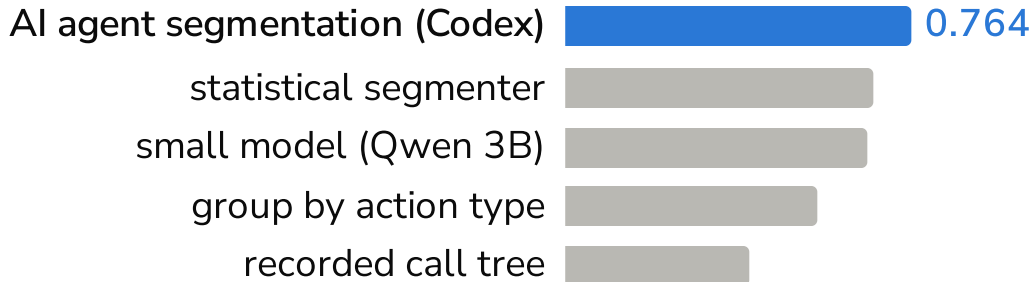
gray: each benchmark's own judge · blue: + AgentPProf profile

Adding the profile raises MAP on all three benchmarks, by 4–56%.

Segments match human stages, at practical cost

DO SEGMENTS MATCH HUMAN STAGE ANNOTATIONS?
(CODETRACEBENCH)

B³ F1 (0–1): for each operation, does its segment match its human stage? Combines precision (no unrelated work merged in) and recall (no stage split apart).



AI agent segmentation matches human stages best; grouping by action type or by the recorded call tree splits each stage into fragments.

WHAT DOES PROFILING COST?

Segmentation runs once

37 min for all 405

CodeTraceBench trajectories (up to 4 workers), about 30K input tokens each

Every later view is cheap

a profile of 27,765 operations builds in **1.16 s**, so each new view or measure takes about a second

Agent observability needs profiling, not only debugging

① Semantic operation stack model

The missing profiling layer for agents: uniform operations and query-time operation stacks, with recursive operation segmentation supplying stable names and nesting.

② AgentPProf

A profiler that turns agent trajectories into pprof-compatible profiles, from prompts down to system effects.

③ Evaluation against hidden ground truth

Public benchmarks and real trajectory datasets: attribution, real problems, human-recognizable structure, practical cost.

Next: profile-driven runtime control, intent-level state-transition models, adaptive triggering, tracing-ecosystem integration.

github.com/eunomia-bpf/agentsight · Questions?