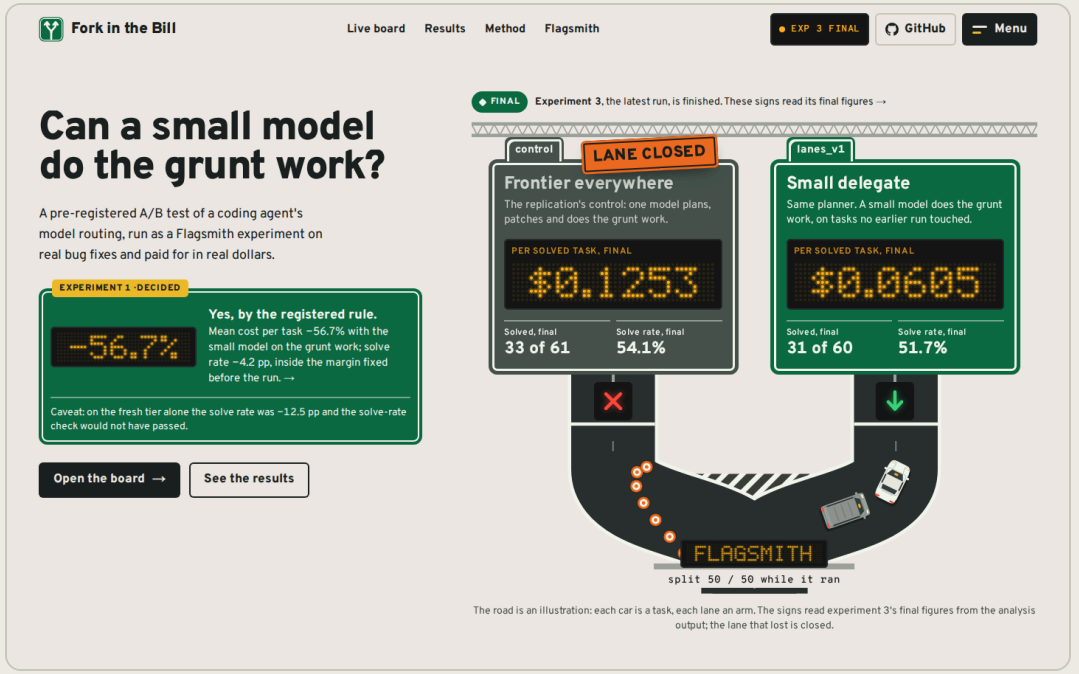
TOGGLE WARS · TRACK 01

Fork in the Bill

Can a small model do a coding agent's grunt work, and cut the cost per solved task?

A pre-registered Flagsmith experiment on real bug fixes, paid for in real dollars.

Fork in the Bill team (CoffeeAndToggle)



THE BILL

Most of an agent's work needs little judgement. It is still billed at the frontier price.

SONNET 5.5 · PER MILLION TOKENS

$2.00 in · $10.00 out

frontier model

HAIKU 5.5 · PER MILLION TOKENS

$0.10 in · $0.50 out

small model, 20x cheaper

WORKER SHARE OF COST, ALL-SONNET ARM

62.5%

Searching the repo, reading long files, running the tests and reading the log: the worker lane. When it runs on the frontier model, that is where most of the money goes.

Prices: `harness/pricing.json`, checked 2026-11-27. Share: experiment 1, `control` arm.

THE QUESTION

Move the worker to the small model. Does each solved task get cheaper, or do fewer tasks get solved?

Decision rule, fixed and hashed before the first task. Ship `lanes_v1` only if all three hold:

1. Flagsmith reports a chance to beat control of at least 95% on `task_cost`.
2. Our Beta(1,1) posterior gives P(Δsolve > −m) ≥ 0.90, with m = 20 percentage points.
3. Flagsmith does not report `task_solved` as losing.

No early stopping. The run ends when every task is done, excluded or invalid. Source: `prereg/experiment-1.md`.

THE DESIGN

Two lanes, one field apart. Flagsmith picks the lane.

`control`

Frontier everywhere

Planner: **Sonnet 5.5**
Worker: **Sonnet 5.5**

`lanes_v1`

Small delegate

Planner: **Sonnet 5.5**
Worker: **Haiku 5.5**

Registered first

`sha256 44d715bd6bd7f8580e8bb1950a21ea852dfe a4daaf040c84009f391bf56e19c2`

Tag `prereg-exp1`, committed 22:57:35 UTC.

Experiment 822 started 23:02:34 UTC.

Amendment 1 added 80 pre-cutoff tasks to clear Flagsmith's 50-per-variant floor, before any per-arm result was seen.

Same prompts, tools, token limits and caps (40 turns, $0.60, 1200 s). The planner has no shell, so all grunt work goes through the worker. Tasks: real SWE-bench-Live issues in 8 languages.

HOW FLAGSMITH IS USED

Flagsmith assigns every task, records it and gives the verdict.

Flag

`agent_routing`, multivariate. Its value is the whole routing policy as JSON, schema-checked on every read.

Variants

Keyed variant `lanes_v1`, plus a `routing_kill` switch that sends no exposure. Never used in the run.

Identities

One task, one identity: `swelive:<id>`, with repo, language and month traits.

Metrics

`task_cost` primary, `task_solved` guardrail, turns and seconds informational.

Experiment

Id 822, 50/50 at 100% rollout, flag locked, prereg hypothesis pasted verbatim.

End and roll out

End Experiment with `lanes_v1` as the decision: it becomes the same flag's value at 100%.

```
# harness/policy.py, once per task
flag = client.get_experiment_flag("agent_routing", identifier=identity, traits=traits)
client.flush_events()   # exposure sent before anything else is tracked
```

THE RESULT · COST

Cost per solved task fell by more than half.

CONTROL · PER SOLVED TASK

$0.1753

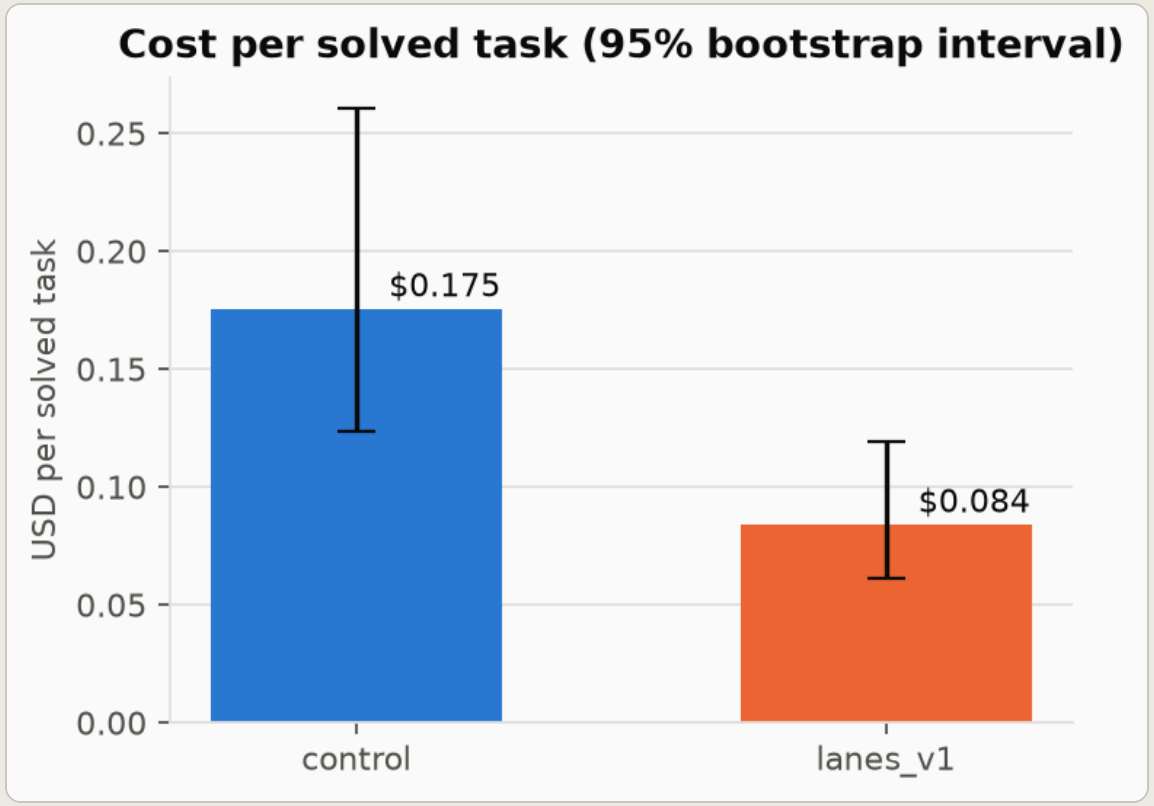
$0.1233 to $0.2607

LANES_V1 · PER SOLVED TASK

$0.0837

$0.0614 to $0.1191

−52.2% per solved task. Mean cost per task $0.0784 → $0.0340 (−56.7%).



76 control and 69 lanes_v1 tasks. Intervals: percentile bootstrap.

Cost per solved task = mean cost ÷ solve rate

THE RESULT · GUARDRAIL

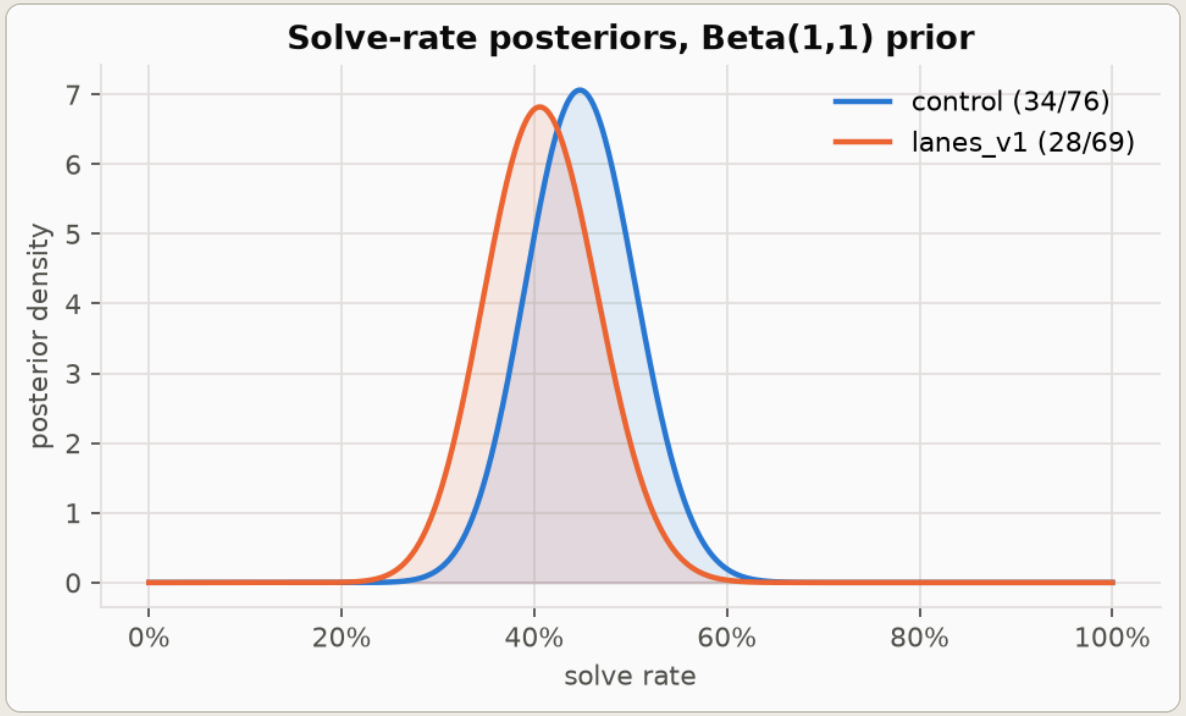
Solve rate: a small drop, inside the margin.

	SOLVED	RATE	INTERVAL
control	**34 / 76**	**44.7%**	34.1% to 55.9%
lanes_v1	**28 / 69**	**40.6%**	29.8% to 52.4%

Δ SOLVE RATE
−4.2 pp
−19.8 pp to +11.8 pp

P(DROP WITHIN M)
97.7%
bar 90.0%, m = 20.0%

This rules out large losses only. On fresh tasks alone the drop is −12.5 pp and the probability is 78.9%, under the bar (secondary analysis).



Beta(1,1) posteriors on solve rate

FLAGSMITH'S VERDICT, THEN OURS

Two methods, one answer: ship `lanes_v1`.

MEAN COST CHANGE	FLAGSMITH (NORMAL)	OURS (BOOTSTRAP)
Point	**−56.6%**	**−56.7%**
Interval	−68.8% to −44.3%	−67.1% to −42.9%
`lanes_v1` mean cost	$0.0341	$0.0340
Sample ratio	p = 0.561, flagged: False	

1. Chance to beat control on cost: **100.0%**. Met: True
2. Posterior within the margin: **97.7%**. Met: True
3. Flagsmith's `task_solved`: **inconclusive**, not losing. Met: True

Flagsmith uses a delta-method normal approximation; we use a bootstrap and Beta posteriors on purpose, so agreement is not an artefact of one method. Results pulled 20261127T120242Z.

INTEGRITY

Don't trust the numbers. Check them.

task_done records	146
Arms re-derived by flag-engine	146 / 146
Env snapshots	56
Distinct env hashes	1
Merkle root	7ed364648c7513ef22280eaed97a2ab77e92d91b8f88f5bbfab08d747a989929

```
# hash chain and Merkle root
uv run python -m integrity.verify_ledger runs/exp1
# every arm recomputed offline
uv run python -m integrity.verify_assignment \
    --run runs/exp1
# registration matches its hash
sha256sum -c prereg/HASHES.txt
# every figure matches numbers.json
uv run python -m analysis.check_consistency
# all of it, every run
make verify
```

AFTER THE DECISION

Rolled out, challenged, then replicated.

DONE

Rollout on the same flag

Experiment 822 ended with `lanes_v1` as the rollout decision. No code changed.

A fresh batch then ran on that flag: 13 tasks, $0.0369 per task, 53.8% solved.

KEPT LANES_V1

Experiment 2: the small model drafts

Registered before its first task. Cost per task −15.4%, solve rate 62.7% to 50.6%, so cost per solved task +4.8%.

The rule said keep `control`, and code applied it in Flagsmith. A negative result, published in full.

SHIPPED LANES_V1 AGAIN

Experiment 3: the same question, new tasks

Registered as a replication. 61 + 60 unused tasks: cost per solved task −51.7% (experiment 1: −52.2%), solve rate −2.4 pp.

Same rule, same answer. Code rolled it out in Flagsmith.

USE THE KIT

Run the same experiment on your own agent.

```
# flag, variant, kill switch, metrics and experiment from your policy files
uv run forkbill init --control flagsmith/policies/control.json \
  --variant lanes_v1=flagsmith/policies/lanes_v1.json \
  --metrics flagsmith/metrics.json --hypothesis-file prereg/experiment-1.md --start
```

```
lanes = RoutedLanes(flagsmith=flagsmith, identity=f"swelive:{instance_id}", pricing=pricing)
reply = client.messages.create(model=lanes.main_model, ...)
lanes.record_call("main", Usage.from_anthropic(reply.usage))
sub = client.messages.create(model=lanes.delegate_model, ...)
lanes.record_call("delegate", Usage.from_anthropic(sub.usage))
lanes.finish(solved=solved, turns=turns)   # task_cost, task_solved, task_turns, task_seconds
```

Python and TypeScript, an OpenFeature adapter, and remote, local, Edge Proxy and edge-local modes. Checked on self-hosted Flagsmith with experimentation on.

LINKS

Read it, run it, check it.

Repository

`github.com/Prashant-thakur77/fork-in-the-bill`

Live site

`forkbill.prashantthakur.dev`

Paper: `docs/paper.pdf`. Every figure in it and in this deck comes from `analysis/out/numbers.json`.

