Real Bugs, Both Solved: The Gap Is Cost

An evaluation of Claude Sonnet 5 (high effort) and Claude Opus 4.8 (high effort) on four decontaminated bug-fix pull requests from production open-source repositories

VulcanBench (github.com/morganlinton/VulcanBench)

Abstract

We evaluate Claude Sonnet 5 and Claude Opus 4.8, both at high reasoning effort, as autonomous coding agents on four bug-fix pull requests merged into production open-source repositories (sqlglot, packaging, more-itertools) between February and June 2026 — after both models' training cutoffs, so every fix is novel. Each task is graded deterministically by the pull request's own failing test; there is no LLM judge. Both models solve all four bugs at 100% (pass@1 over three repeats, 24 runs total), including a subtle set-algebra bug in version-range policy. Correctness therefore carries no information about model choice on this suite. Cost does: the same answers differ by 3.4× in total spend ($19.88 versus $5.89), and the gap is concentrated almost entirely in the 76k-line repository, where Sonnet 5 spends $15.15 per task against Opus 4.8's $2.08 — a 7× difference driven by navigation behavior, not price. This pilot established the design principle carried through every subsequent VulcanBench report: once tasks are real and decontaminated, economics is a first-class result.

1. Introduction

Public coding benchmarks saturate quickly: when every frontier model scores 98–100%, the benchmark measures its own ceiling rather than the models. The usual response is to author harder puzzles. This report takes a different route: keep the tasks real — actual bugs, actually fixed, in actually deployed libraries — and measure what the models spend to close them. If correctness converges, the spend becomes the signal.

We ask the narrowest version of the question: on four real, decontaminated bug fixes that both models can complete, what separates two frontier agents from the same vendor? The answer is not accuracy, refusals, or latency. It is cost, and the shape of the cost is informative: it rides on repository scale.

2. Method

2.1 Task construction

Each task is a real merged pull request. The repository is sliced at the pull request's base commit; the agent receives a terse issue describing the symptom — as an engineering team would hand off a ticket — plus the full working tree. It must produce a patch. The patch is graded by the pull request's own test: validated to fail on the base commit and pass on the merged fix. Grading is deterministic; no model judges another model.

2.2 Decontamination

All four pull requests were merged between February and June 2026, after both models' training cutoffs. The fix cannot be recalled from memory; it must be derived from the code.

2.3 Configuration

Both models run at high reasoning effort in a network-isolated Docker sandbox with identical step budgets and tooling. Three repeats per task per model: 2 models × 4 tasks × 3 repeats = 24 runs, $25.77 total spend. Costs use list prices (Sonnet 5: $3/$15 per million input/output tokens; Opus 4.8: $5/$25).

Table 1. The four tasks. Each is a real merged pull request; the hidden test grades the stated requirement.

Task	Repository	What the agent must do	Scale
`interleave-empty`	more-itertools	Fix interleave behavior on empty iterables	small
`packaging-empty-name`	pypa/packaging	Reject empty distribution names in requirements	small
`versionrange-policy`	pypa/packaging	Correct VersionRange prerelease set-algebra policy	small
`sqlglot-parser-fix`	tobymao/sqlglot	Fix a parser bug in a 76k-line SQL transpiler	76k LOC

Table 2. Aggregate results. Both models solve every task in every repeat; costs are per-task means over three runs.

Model (effort)	Score	pass@1	Total cost	$/task	76k-LOC task
Claude Sonnet 5 (high)	4/4	100%	$19.88	$4.97	$15.15
Claude Opus 4.8 (high)	4/4	100%	**$5.89**	**$1.47**	**$2.08**

Figure 1. Cost per task at equal (100%) correctness. Dark blue: Sonnet 5 (high). Rust: Opus 4.8 (high). The gap is concentrated in the 76k-line repository.

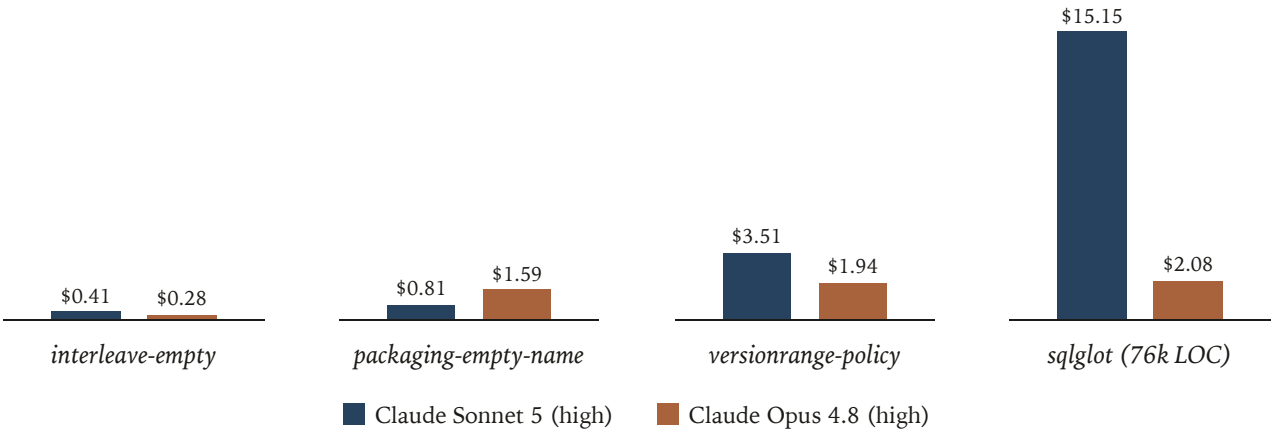

Table 3. Per-task cost (USD, mean of three runs). Every cell is a pass; no run failed and no run was refused.

Task	Sonnet 5 (high)	Opus 4.8 (high)	Ratio
`interleave-empty`	$0.41	$0.28	1.5×
`packaging-empty-name`	$0.81	$1.59	0.5×
`versionrange-policy`	$3.51	$1.94	1.8×
`sqlglot-parser-fix`	$15.15	$2.08	7.3×
Total	**$19.88**	**$5.89**	**3.4×**

4. Findings

4.1 Correctness is saturated

Both models solve every real, decontaminated bug at 100%, including the subtle VersionRange set-algebra policy fix, in every repeat. Functional pass/fail no longer separates the frontier on ticket-sized work; a benchmark that reports only accuracy would score these models identical.

4.2 Cost is the live axis

The same answers cost 3.4× more on one model than the other: $19.88 versus $5.89 for the full suite. Neither list price nor token price explains the gap; it is dominated by how much work each agent does before converging.

4.3 The gap rides on repository scale

On the three small tasks the models are within ordinary variance of each other — Sonnet 5 is cheaper on one, Opus 4.8 on two, and the absolute differences are cents to a few dollars. On the 76k-line sqlglot repository the difference is structural: $15.15 versus $2.08 per task, a 7× gap. Sonnet 5 thrashes while navigating the large codebase — re-reading files and re-deriving context — where Opus 4.8 stays efficient. Repository scale, not task difficulty, is what separates these agents.

4.4 The gap flips by scale

The direction is not fixed: on small tasks Sonnet 5 is frequently the cheaper model (a pattern later confirmed by the 936-run effort sweep of Report No. 2), while on large-repository navigation Opus 4.8 wins decisively. Model selection on this workload is a function of the codebase, not just the model.

5. Discussion

This pilot inverts the usual benchmark question. When two frontier models from the same vendor land identical pass rates on real work, per-task correctness has stopped carrying information; what differs is what each model spends to get there, and where. For teams selecting a model for ticket-sized engineering, the operative questions are cost per solved task and how that cost scales with repository size — not leaderboard accuracy.

The result also sets the benchmark's roadmap. Because the frontier saturates real bug-fix work, VulcanBench treats economics — cost, wall-clock time, agent steps, tokens — as first-class results alongside accuracy in every subsequent report, and sources tasks that keep at least one frontier model failing (Report No. 4's `flask-teardown-robust` descends directly from this principle).

6. Limitations

Four tasks is a pilot, not a survey; the 3.4× total is dominated by a single large-repository task, and we report it as such. Costs use list prices and exclude provider-side caching discounts (later reports fold these in). Three repeats bound run-to-run variance but not model-version drift.

7. Reproducibility

Every task, hidden test, gold patch, per-task Docker image, and the grading harness are open source. Each run writes a full trace, summary, and replayable HTML transcript. Reproducing this report costs approximately $26 at list prices: `vulcanbench run --suite v2-pilot --model <model> --effort high --repeat 3`.