Grok 4.6 in Cursor vs. a bare-bones harness

The first model-times-harness study on VulcanBench: one model, two delivery systems, on twenty-three real merged post-cutoff PRs. The bare-bones system is Report No. 14; the Cursor system is new, three attempts per task at four effort levels.

Abstract

The harness is worth more than the effort knob: at the same nominal *high* setting, Grok 4.6 scores 94.9% inside Cursor against 73.9% through VulcanBench's deliberately minimal reference loop, a 21.0-point gap. Through the raw API the model's effort curve peaks at medium and collapses; inside Cursor it rises monotonically (79.7 → 94.2 → 94.9 → 95.7) and tops out at xhigh. The overthinking failure mode disappears entirely: the raw API produced 10 budget-cut runs above low effort, Cursor produced zero at any level. The harness also changes which tasks are solvable, not just how many. The two tasks no raw-API run has ever solved in Reports No. 12–14 both fall inside Cursor, while a task every raw-API level solves becomes Cursor's only recurring failure. The same nominal level runs 1.6–4.9 times faster inside Cursor. Only at low effort does the bare-bones loop win, by 2.9 points.

Table 1. The two systems

	System A: bare-bones	System B: inside Cursor
Path	xAI API, VulcanBench minimal reference loop	cursor-agent CLI, Cursor's own scaffold
Effort control	reasoning_effort sent per request	baked into the model id (cursor-grok-4.6-low … -xhigh)
Observability	tokens, cache reads, dollars at list price	no token or cost reporting (Cursor credits)
Attempts	1 per task (Report No. 14, $52.70 recorded)	3+ per task, all 23 tasks, four levels

Table 2. pass@1 by nominal effort level (mean per-task success rate; ±1 stderr)

Effort	In Cursor	Bare-bones	Delta	Cursor time/task	Bare-bones time/task
low	79.7% ±8.1	82.6% ±8.1	−2.9	2.9 min	4.8 min
medium	94.2% ±4.4	87.0% ±7.2	+7.2	2.9 min	14.2 min
high	**94.9% ±4.4**	**73.9% ±9.4**	**+21.0**	**4.2 min**	**16.3 min**
xhigh	95.7% ±4.3	78.3% ±8.8	+17.4	4.2 min	15.8 min

Cursor columns: 84/84/79/75 runs (three attempts per task; a fifth of the low and medium runs are recovered attempts re-verified in Docker after a harness fix). Bare-bones columns: 23 runs each, single attempt. Times are medians; runtime includes provider latency.

Findings

1. The harness is worth more than the effort knob. The largest gap the knob produced in Report No. 14 was 13.1 points (medium vs high). The harness produces 21.0 at the same nominal setting, and every Cursor level from medium up beats every bare-bones level.

2. The overthinking failure mode disappears. Through the raw API, failures above low effort are dominated by runs cut off at a step or wall-clock budget (2, 4, and 4 at medium, high, xhigh). Inside Cursor there are zero budget-cut runs at any level, and the same nominal level finishes 1.6–4.9 times faster.

3. The harness changes which tasks are solvable. *pennylane-trotter-fragmented* and *sqlglot-canonicalize-internal-names*, unsolved by any raw-API run across Reports No. 12–14, are both solved inside Cursor. In the other direction, *itertools-strip-prefix*, solved at every raw-API level, fails at medium, high, and xhigh inside Cursor.

4. Low effort is the exception that sharpens the question. The bare-bones loop wins low by 2.9 points. Whatever Cursor's scaffold adds, it pays off where the model reasons long and costs a little where it does not.

5. This is a product measurement, not a weights measurement. Read it as "Grok 4.6 as Cursor ships it" against "Grok 4.6 as the API hands it to you." The 21-point delta is far outside both columns' uncertainty; its cause is not identifiable from outside (see Caveats).

Caveats

The gap could come from Cursor's scaffold (context management, tool design, stopping rules), from how Cursor's effort-suffixed model ids map to the API's reasoning_effort values, or from different serving. "Cursor Grok 4.6" is Cursor's own branding and may be a tuned or differently-hosted deployment; these explanations are indistinguishable from outside, and per-request token counts, which Cursor does not report, would settle it. Attempt counts differ between systems (3 vs 1), so per-task flips carry more weight on the Cursor side. Bare-bones costs in Report No. 14 are lower bounds: xAI reports reasoning tokens outside completion_tokens, and the harness under-counted them at the time of that sweep (since fixed). Cursor-side spend is not reportable by the CLI and appears only in the account dashboard.

Method

VulcanBench v3: 23 tasks from real merged post-cutoff PRs (Python 9, TypeScript 4, Rust 4, Go 3, JavaScript 3). Both systems are graded by the same deterministic hidden tests in Docker; model judges disabled. System A ran 2026-08-12 (Report No. 14); System B ran 2026-08-13/14 with cursor-agent 2026.08, non-fast model ids, Cursor sandbox enabled, and VulcanBench setup and verification in Docker over the agent's host workspace. pass@1 is the mean per-task success rate, so uneven attempt counts do not bias it. This is the first VulcanBench study measuring one model through two harnesses; the series continues with other model-times-harness combinations.