Qwen3.8-Max across the effort knob

First measurement of Alibaba's Qwen3.8-Max on the full v3 suite: twenty-three real merged post-cutoff PRs across five languages, at low, medium, and xhigh effort, three attempts per task.

Abstract

Qwen3.8-Max's effort knob runs backwards, further than any model measured here: 81.2% at low, 71.0% at medium, 55.1% at xhigh — a 26-point fall to the lowest column on the v3 board. Since xhigh is the API default, an untuned integration gets the worst setting. The deficit is unfinished work, not bad work: wrong answers vanish (3 → 0 → 0) while runs that never finish climb (14% → 29% → 41%), and six tasks low solves 3-for-3 account for 83% of the drop. At $126.25 against DeepSeek V4-Flash's $13.60 on the same suite, and 19.6–25.5 min/task, it is the most expensive and slowest model on the board — and the first debut measured at three attempts per task.

Table 1. Results (three attempts per task per effort level)

Effort	pass@1	Solved	Wrong	Unfinished	Cost	Tok/task	Time/task	Steps	$/task	$/solved
low	**81.2%**	**56/69**	**3**	**10**	**$42.09**	**188 K**	**19.6 min**	**179**	**$0.61**	**$0.75**
medium	71.0%	49/69	0	20	$38.81	157 K	21.7 min	152	$0.56	$0.79
xhigh *(default)*	55.1%	38/64	0	26	$45.35	208 K	25.5 min	167	$0.71	$1.19

pass@1 is the mean per-task success rate across attempts, so uneven counts do not bias it. Wrong = a finished run that failed the hidden tests; unfinished = cut off at the wall-clock or step budget, unchanged from Reports 07, 08 and 10. The xhigh column has 64 runs, not 69: five died on 600-second API read timeouts (below), excluded rather than scored zero. Qwen's enum is low/medium/xhigh — no high — xhigh being the default.

Findings

1. The knob runs backwards, and steeply. Low leads xhigh by 26 points, more than triple the 9-point inversion Report 10 recorded for Claude Opus 5, and far outside the noise: standard errors are ±7.8, ±9.4 and ±9.7 points, and low-vs-xhigh does not overlap. Paying for effort is the difference between mid-table and last.

2. Effort converts wrong answers into unfinished runs. Wrong answers fall to zero (3 → 0 → 0) while runs that hit a budget climb (10 → 20 → 26, or 14% → 29% → 41%). At medium and xhigh *every single failure* is an incomplete run. Failed runs burn ~5× the completion tokens of successful ones (115–140 K vs 22–29 K) and 3× the clock — not sloppy attempts, but runs cut off mid-work.

3. The regression is in solvable work. Three tasks score zero at every setting; extra reasoning rescues none of them. Nine regress from low to xhigh, and the six that low solves 3-for-3 account for 83% of the drop, three collapsing to zero. It is not losing the hard problems. It is losing the ones it already knows how to do.

4. Most expensive, slowest, and beaten on every axis. The sweep cost $126.25 against $13.60 for DeepSeek V4-Flash on the identical suite. Every DeepSeek column (88.4 / 87.3 / 85.5) outscores every Qwen column (81.2 / 71.0 / 55.1) with no overlap, at $0.08 per solved task against $0.75–1.19. At 19.6–25.5 min/task it is the slowest model on v3, ahead of Kimi K3 at 17.2.

5. A second clock, imposed by the provider. Five xhigh runs on the suite's heaviest repositories failed with 600-second read timeouts from the DashScope endpoint — a single request exceeding ten minutes, well inside the harness's own 45- and 60-minute budgets. No previous report has recorded this. They are logged as run *errors* rather than scored zero, the treatment safety refusals receive, and retrying reproduced them: at its default effort, the model can exceed its own provider's response window.

Table 2. Failure map (the nine tasks that moved; fourteen scored identically at all three settings)

Task	low	medium	xhigh
jiff-strftime-negpad	3/3	3/3	0/3 — 3 unfinished
semver-inc-dotted-prerelease	3/3	0/3 — 3 unfinished	0/3 — 3 unfinished
sqlglot-qualify-lateral-star	3/3	1/3 — 2 unfinished	0/3 — 3 unfinished
packaging-range-prerelease-policy	3/3	3/3	1/3 — 2 unfinished
semver-xrange-order	3/3	3/3	1/3 — 2 unfinished
zod-invert-codec	3/3	3/3	1/3 — 2 unfinished
itertools-strip-prefix	3/3	3/3	2/3 — 1 unfinished
flask-teardown-robust	1/3 — 2 unfinished	0/3 — 3 unfinished	0/3 — 3 unfinished
networkx-leiden-communities	1/3 — 2 unfinished	0/3 — 3 unfinished	0/1 — 1 unfinished

Never solved at any setting: aiohttp-upgrade-deferred, pennylane-trotter-fragmented, sqlglot-canonicalize-internal-names. Cells are solved/attempted. Of the 56 unfinished runs across the sweep, 38 died at the 45-minute (large) or 60-minute (xlarge) wall-clock boundary and only three hit the step ceiling; none was cost-capped. The clock, not the step budget, is what binds.

Method & caveats

VulcanBench v3: 23 tasks from real merged post-cutoff PRs (Python 9, TypeScript 4, Rust 4, Go 3, JavaScript 3), network-isolated Docker sandboxes, hidden-test grading. Budgets scale with repository size — 50–200 steps, 5–60 min — identical for every model and unchanged since Report 07. Priced at $2/$6 per M tokens, international (Singapore) endpoint. Caveats: with an unlimited budget xhigh might close some of the gap, so this measures capability under a fixed budget, not a ceiling; Qwen's `thinking_budget` cap — the knob most likely to address this pattern — was untested because its API rejects it alongside `reasoning_effort`. Every column here is three attempts per task; the single-attempt columns of Reports 07, 08 and 10 carry wider uncertainty.