Grok 4.5 vs Claude Fable 5 vs GPT-5.6 Sol

Twenty-three real merged post-cutoff PRs across Python, Rust, TypeScript, JavaScript, and Go, graded by deterministic hidden tests.

Abstract

Grok 4.5 leads at 91% from medium effort onward -- high effort buys nothing but a 31% larger bill. GPT-5.6 Sol is the only model whose accuracy climbs monotonically with effort (78 to 83 to 87%); Fable 5 oscillates at 85+/-2% regardless of the knob. The two frontier-hard tasks went 0-for-27. Total benchmark spend $96.72.

Table 1. Results

Model (effort)	Score	pass@1	Cost	Tok/task	Time/task	$/solved
Grok 4.5 (medium)	**21/23**	**91.3%**	**$6.67**	**106 K**	**3.9 min**	**$0.32**
Grok 4.5 (high)	21/23	91.3%	$8.76	141 K	4.9 min	$0.42
Claude Fable 5 (low)*	20/23	87.0%	$12.18	28 K	3.4 min	$0.61
GPT-5.6 Sol (high)	20/23	87.0%	$15.90	85 K	4.2 min	$0.80
Claude Fable 5 (high)*	20/23	87.0%	$20.82	44 K	4.3 min	$1.04
Grok 4.5 (low)	19/23	82.6%	$3.39	57 K	2.5 min	$0.18
GPT-5.6 Sol (medium)	19/23	82.6%	$8.83	50 K	2.8 min	$0.46
Claude Fable 5 (medium)*	19/23	82.6%	$16.32	36 K	3.7 min	$0.86
GPT-5.6 Sol (low)	18/23	78.3%	$3.85	23 K	1.5 min	$0.21

** Fable 5 ran with Opus 4.8 refusal fallback: 6 of 69 runs (8.7%) were declined by the API safety classifier (category 'cyber'; same three tasks at every effort) and served end-to-end by Opus 4.8; all six passed. Tokens are billed tokens (prompt-cache reads discounted); time is sandbox wall-clock.*

Findings

1. Grok plateaus at medium.
High effort spends 33% more tokens and 31% more money for the identical 21/23, failing the same two tasks. No config beats 91.3% at $0.29/task.

2. Only Sol rewards the effort knob.
78.3 to 82.6 to 87.0%, monotonic. Fable oscillates at 85+/-2% with a different failure set each run -- the knob changes which borderline tasks fall, not how many.

3. Frugal tokens, premium price.
Fable uses the fewest tokens at every effort (28-44 K/task) yet is the costliest per solved task ($0.61-$1.04) at $10/$50-per-M pricing.

4. The difficulty ceiling held; one wall fell.
PennyLane Trotter fragmentation and SQLGlot internal-name canonicalization went 0-for-27. Flask teardown redesign fell to all three models -- but only at high effort.

Failure map (tasks failed by at least one config)

Task	Grok l/m/h	Fable l/m/h	Sol l/m/h
pennylane-trotter-fragmented	x x x	x x x	x x x
sqlglot-canonicalize-internal-names	x x x	x x x	x x x
flask-teardown-robust	x ok ok	x ok ok	x x ok
aiohttp-upgrade-deferred	x ok ok	ok* ok* ok*	x ok ok
itertools-strip-prefix	ok ok ok	ok ok ok	x x x
networkx-leiden-communities	ok ok ok	ok x x	ok ok ok
sqlglot-iso8601-nanos	ok ok ok	ok x ok	ok ok ok

*Remaining 16 tasks: all passed everywhere. * Fable cells marked ok* were served by the Opus 4.8 refusal fallback.*

Method & reproducibility

Suite v3: 23 tasks from real merged post-cutoff PRs (Python 9, Rust 4, TypeScript 4, JavaScript 3, Go 3). Network-isolated Docker sandboxes; deterministic hidden-test grading; one attempt per task per effort (low, medium, high). Every task pre-validated: gold patch = 1.0, unpatched = 0.0, deterministic over 3 runs. Pricing: Grok 4.5 $2/$6, GPT-5.6 Sol $5/$30, Claude Fable 5 $10/$50 per million tokens. Reproduce: vulcanbench run --suite v3 --model <provider:model> --effort <low|medium|high> --no-judges