

Kimpton CostBench: Measuring Task Outcomes and Resource Use in Tool-Using Systems

Kimpton

October 3, 2026

Abstract

Kimpton CostBench evaluates complete tool-using configurations by linking objective task outcomes to versioned execution records, resource receipts, and separately declared prices. We execute 450 attempts across 30 synthetic software-repair, database, and simulated API tasks, grouped into 11 templates. Three model-and-harness configurations each receive five repeats per task. A pre-dispatch all-attempt protocol retains every selected attempt, including ten infrastructure or execution failures. GPT-4.1 achieves 146/150 successes, GPT-4.1 mini 132/150, and Claude Haiku 4.5 140/150. All 890 execution and grading sandbox resources have proven terminal measurements. Repricing inference usage and sandbox CPU and memory under frozen published rates gives measured cost per success of \$0.00548, \$0.00230, and \$0.01034, respectively. GPT-4.1 and mini form the observed quality-cost frontier. Paired template-cluster intervals distinguish the measured cost-per-success differences but do not resolve success-rate differences. Exact no-dispatch audits establish zero inference use for nine otherwise incomplete receipts. The release preserves failed-attempt accounting, price provenance, operational recovery amendments, and offline analysis code. Published-rate costs and synthetic-credit ledger debits are separate from invoices and total operating cost. The evidence describes this small synthetic workload and its recorded execution conditions.

1 Introduction

A tool-using system can make several model requests, call tools, retry an operation, modify state, and use a sandbox before a task receives a grade. Token totals alone describe only part of that execution. A model label also omits the harness, prompts and policies, tool interface, runtime, and resource limits that shape task outcomes. Comparisons are difficult to interpret when task success, complete system identity, resource observations, and the basis for a monetary calculation cannot be inspected together.

We present Kimpton CostBench, an implemented synthetic benchmark workload and evidence protocol for comparing complete tool-using configurations on software repair, database operations, and simulated API state changes. For each attempt, the record binds the selected task and repeat to a system configuration, episode outcome, independently checked task result, and available resource receipts. Raw quantities, prices, billing projections, and telemetry are represented as separate fields. This makes missing evidence visible and allows later calculations to use a declared price vector without changing the underlying observations.

The benchmark name has a direct prior collision. Liu et al. use the title *CostBench* for a benchmark of cost-optimal multi-turn planning and adaptation in dynamic travel environments [4]. Their work is direct prior art for cost-aware tool-agent evaluation. Kimpton CostBench studies a different workload and focuses on evidence linking task outcomes with execution resource records.

We do not claim naming priority or novelty for cost-quality assessment, cost frontiers, state-based grading, repeated trials, or execution-based coding grades.

The contributions are: (1) an authored 30-task workload across three task families and 11 templates; (2) an implemented protocol connecting objective task outcomes to versioned resource evidence with explicit provenance and coverage; and (3) a complete 450-attempt empirical record with outcome and measured-resource cost comparisons, coverage disclosures, and reproducible derived analysis. Complete coverage of this declared priced-resource subset does not establish total operating cost. Conclusions apply only to the frozen tasks, configurations, execution controls, and price basis.

2 Related work

Cost-aware agent evaluation predates this benchmark. Liu et al.’s existing CostBench evaluates multi-turn travel planning with atomic and composite tools that have configurable costs, in static and dynamic settings with blocking events and replanning [4]. It measures cost-optimal planning and adaptation. The present workload instead covers software repair, database changes, and simulated API state changes, and its accounting design connects task outcomes to execution receipts and declared price calculations. This difference in workload and evidence scope does not imply superiority.

Kapoor et al. argue for evaluating agents with cost and accuracy together, using controlled budgets and cost-quality tradeoffs, and discuss holdout design and reproducibility [2]. We therefore do not present joint quality and cost evaluation, or a Pareto view, as a new idea. HELM evaluates models over multiple scenarios and metrics, including efficiency measures and coverage reporting [3]. Multi-metric evaluation and efficiency measurement are established practice.

The task grading also builds on established approaches. τ -bench evaluates tool-agent interactions against final state and response requirements and reports repeated-trial reliability [5]. SWE-bench evaluates generated changes through repository tests on real GitHub issues [1]. CostBench uses independent state checks for database and API tasks and functional tests for its small authored software-repair tasks. Objective state grading, repeated trials, and functional testing are not claimed as novel. Compared with SWE-bench, this task set is much smaller and synthetic; it does not aim to represent the breadth of real repositories.

Table 1 summarizes this positioning. The proposed distinction is an implementation-level combination of these established elements with versioned resource receipts and explicit economic categories, with explicit limits on the released evidence.

Table 1: Prior work and the bounded scope of this manuscript.

Work	Established scope relevant here	Scope in Kimpton CostBench
Liu et al. [4]	Cost-optimal tool planning and adaptation in travel tasks with configurable tool costs.	Synthetic operational tasks; execution-resource evidence linked to graded outcomes.
Kapoor et al. [2]	Agent accuracy-cost evaluation, budgets, Pareto comparisons, and reproducibility concerns.	Same decision dimensions, with a specified receipt and price-provenance protocol.
HELM [3]	Broad multi-scenario, multi-metric model evaluation including efficiency.	Three authored task families and per-attempt evidence records.
τ -bench [5]	Tool-agent interaction, final-state grading, and repeated-trial reliability.	Database and simulated API final-state graders plus software functional tests.
SWE-bench [1]	Repository issue repair graded through execution and tests.	Smaller synthetic code-repair tasks, not real-repository coverage.

3 Benchmark workload and controls

3.1 Task families

The implementation defines 30 generated synthetic tasks, divided evenly among software repair (SWE), database operations, and simulated API state changes. Task identifiers and the generator’s kinds show three SWE templates (invoice rounding, inventory reservation, retry idempotency), four database templates (insert reconciliation, inventory update, ledger report, migration backfill), and four API templates (deployment rollout, ticket reconciliation, access policy, quota update). Thus there are 11 shared templates, not 30 independent task families. Private seeded values create variants within these templates. The scored inputs and expected answers remain in a private source tree.

SWE tasks provide a small Python module and a prompt to repair a function. The implemented task kinds cover half-up cent rounding, rejecting invalid inventory reservations, and counting distinct completed operation identifiers. The grader does not execute submitted code in its own process. It requires a candidate-test receipt with the expected protocol, candidate digest, successful exit, bounded execution record, and outputs for all hidden cases. Before the campaign, the isolated candidate runner passed correct/incorrect-case, reset, replay, and state-isolation qualification for the scored task families.

Database tasks begin from task-specific SQLite rows and ask the system to perform an operation through the bounded `db_query` tool. The task kinds cover idempotent upsert, inventory adjustment, aggregation into a report table, and migration backfill. The guest initializes isolated task state. The authoritative grade uses root-collected final SQLite rows and compares them with protected expected rows. An agent-authored completion artifact does not establish success.

API tasks use a local simulator, never a real external service. The task kinds cover deployment version update, exactly-once ticket resolution, role update while preserving other members, and quota change while preserving current use. The grader uses root-collected final simulator resources and checks them against protected expected state. API action arguments and task answers are not needed in public receipt projections.

3.2 Execution protocol and controls

The adapter declares the protocol identifier `evalrouter.costbench.v1`, with family-specific tool schemas. SWE exposes `read_file`, `write_file`, `run`, and `submit_source`; database exposes `db_query`; API exposes `api_request`. The runtime manifest binds hashes for the guest, candidate runner, bridge, metering module, tool schemas, task set, grader, dependencies, and environment and grader images. Runtime validation rejects mismatched hashes, family identities, tool schemas, or unqualified images.

Declared execution ceilings include at most 1,000 CPU-millis, 2,048 MiB memory, 300 seconds duration, 20 steps, 32,768 observation bytes, and blocked network. The package contract specifies a 30-second command limit; SQLite statements are bounded to two seconds, 100 result rows, and 32,768 result bytes. The qualified isolation contract uses distinct Linux users and a seccomp policy without process forking. These are configured constraints; the resource counters described later do not by themselves prove that all physical resource use is observed.

A comparison treats the system as a full model-and-harness configuration. The frozen campaign names three provider routes: OpenAI `gpt-4.1-2025-04-14`, OpenAI `gpt-4.1-mini-2025-04-14`, and Anthropic `claude-haiku-4-5-20251001`. The corresponding tool-qualified route identifiers are recorded in the private campaign manifest. Route identity does not fully specify the harness, prompts, policies, retry behavior, sampling, family-specific execution identity, or runtime. The frozen systems document records complete configurations and their matching fingerprints. Model route names alone do not identify these configurations.

Each configuration receives the same 30 task IDs and five repeat indices, yielding 450 selected attempts. The campaign contains 45 family-by-configuration-by-repeat runs, each selecting ten tasks. All 45 native runs reach terminal reports. Their immutable reports account for all 450 attempts: 440 completed attempts, eight infrastructure failures, and two execution failures, with no replacement attempts, cancellations, or budget skips.

CostBench evidence and analysis flow



costbench-v1 CostBench@costbench-v1 · formula costbench-v2-all-attempts · prices costbench-published-2026-10-02-v1 basis
 dated 2026-10-02 · system prices gpt-4.1=costbench-gpt-4.1-published-2026-10-02-v1,
 gpt-4.1-mini=costbench-gpt-4.1-mini-published-2026-10-02-v1,
 claude-haiku-4.5=costbench-claude-haiku-4.5-published-2026-10-02-v1 · source
 98e6afe2787a8e498383d7d2fac00a38198c6b000bc4987a2ce3725785c13772 · 2026-10-03 · systems: gpt-4.1, gpt-4.1-mini,
 claude-haiku-4.5

Figure 1: Source-generated execution and evidence path. Every derived table and figure is bound to the same sanitized analysis input.

4 Outcome and resource accounting

4.1 Outcome definition

The analysis protocol freezes formula version `costbench-v2-all-attempts`, a higher-is-better score direction, success threshold $Q \geq 1.0$, and measured priced resources as the cost scope. Task utility Q is the grader reward in $[0, 1]$ when available. Success is binary and requires the threshold and required state or safety checks. The task-specific score does not represent monetary business value.

Every selected attempt remains in the primary outcome denominator, including infrastructure errors, grader failures, task timeouts, model errors, failed actions, and budget exhaustion. The protocol was frozen before dispatch. Failed attempts are not replaced. Missing grades count as unsuccessful outcomes, while the source retains their unavailable utility and recorded status. Resource accounting includes every attempt. Missing resource quantities remain unknown.

4.2 Receipt identities, coverage, and unknowns

The economic receipt projection reads canonical inference-call, episode, and episode-operation records. It applies identity-based deduplication and refuses conflicting duplicate receipts. Its privacy-minimized projection omits prompts, task payloads, action arguments, responses, grader answers, endpoint details, and credentials. The assembler binds scoped immutable reports to campaign run identifiers, selected task/repeat bindings, parent system fingerprints, family fingerprints, and price data. It validates receipt checksums and does not fill missing quantities or calculate charges. A checksum supports offline integrity checking; source provenance from the scoped report export is represented separately.

Resource observations are typed quantities with units, source, and task/run/system linkage. The priced scope consists of provider-reported inference-token quantities and provider-reported sandbox CPU, memory, and GPU counters. The sandbox configurations request CPU and memory resources without a GPU. Their GPU counters do not measure hardware used behind the managed inference endpoints. Descriptive diagnostics include provider dispatches, tool calls, database calls, API calls, additional dispatched attempts, and episode elapsed time when available. Provider dispatch count is distinct from logical model steps. These diagnostics are not part of the priced-resource subtotal. Elapsed intervals can overlap and are not added as resource cost. Episode elapsed time uses the recorded creation and completion timestamps. For the CostBench grading path, completion precedes terminal-resource cleanup and counter collection. This measures time to recorded outcome and does not include the subsequent meter collection window or establish worker throughput. Missing values are unknown, not zero. Exact repeat operations are deduplicated by durable identity; conflicting reuse of an event or receipt identity invalidates the evidence.

4.3 Price bases

Resource quantities are separate from a versioned price vector. The assembler permits explicit prices for input, cached-input, output, total-output, reasoning-output tokens and CPU, memory, or GPU time. Each price vector declares a name, URL, source digest, effective date, and version. The observation date of a retrieved price page is distinguished from the provider’s effective date when the latter is unavailable. Token rates may differ by configured model; CPU and memory rates are recorded in the common base vector. The vector used for analysis is frozen independently from a route’s historical declaration. CPU and memory counters are recorded in CPU core-nanoseconds and GiB-nanoseconds before conversion to priced units.

Token categories are mutually exclusive for pricing. When cached-input usage is known as a subset of prompt input, the uncached input quantity is prompt input less cached input, and the cached quantity uses its own rate. If that split is unavailable, it is not inferred from aggregate token counts. A separately bound route contract may establish a structural zero for an unsupported category; the original receipt and the contract basis remain distinct. Output and reasoning-output partitions are priced separately only when the receipt reports the split. When only authoritative aggregate completion usage is available, that total is priced once and never added to output or reasoning partitions. A priced aggregate does not establish a known reasoning-token count.

For standard OpenAI and Anthropic routes, the immutable ledger field `provider_cost` is calculated using provider-reported token usage and route-declared prices. It is a configured-price accounting estimate, not a provider-reported dollar charge or invoice. Any independently frozen analysis-vector estimate is a separate calculation. The field `customer_billed_micros` records a debit in a local test ledger funded with synthetic credits. It is not collected revenue, evidence of cash payment, or an issued customer invoice. No service-fee revenue is established by this field. These ledger debits are separate from provider estimates and sandbox resource repricing. Provider-reported sandbox CPU, memory, or GPU counters are telemetry; they are not independent invoice reconciliation. These counters describe the provider’s sandbox resource accounting, rather than an independently sampled process CPU profile. A fixed compute tariff is a quoted customer price, not a measured physical compute cost. Unpriced resources stay unknown, so a complete total economic cost cannot be asserted when material quantities or rates are unavailable.

The principal release analysis summarizes a measured priced-resource subset. It excludes storage, network transfer, external human work, external paid tools or services, platform overhead, and reset, grading, or orchestration costs where those quantities are not measured and covered by the frozen vector. A configured provider-price estimate, local synthetic-credit ledger debit, and recomputed sandbox-resource subtotal are separate measures and are never added without a documented non-overlap rule. The priced subset is not total cost. Cost per success is undefined when there are no eligible successes, and is withheld as a comparative measure when any required measured-scope quantity or price is unresolved for any attempt. Known priced subtotals remain reportable with missing categories disclosed. Price revisions form new analyses and do not mutate historical receipts.

4.4 Metrics and denominators

For configuration s , let A_s contain every selected attempt and let $E_s = A_s$. Let u_i be the grader utility. Unavailable utility is represented as zero only when computing the eligible-outcome mean; it is imputed in source evidence. Let $y_i = \mathbf{1}[u_i \geq 1]$ when utility is known, and zero otherwise. The outcome summaries are

$$\hat{p}_s = \frac{\sum_{i \in E_s} y_i}{|E_s|}, \quad \bar{u}_s = \frac{\sum_{i \in E_s} u_i}{|E_s|}. \quad (1)$$

They are undefined for an empty eligible set. For resource category r , a deduplicated quantity x_{ir} and frozen configuration-specific unit price p_{sr} yield a known priced subtotal

$$c_i^{\text{known}} = \sum_{r \in K_i} x_{ir} p_{sr}, \quad \bar{c}_s = \frac{1}{|A_s|} \sum_{i \in A_s} c_i, \quad \hat{C}_{\text{success},s} = \frac{\sum_{i \in A_s} c_i}{\sum_{i \in E_s} y_i}. \quad (2)$$

Here K_i contains only known and priced quantities. Complete per-attempt measured-scope cost c_i is defined only when every required category is measured and priced; otherwise it is unknown while

c_i^{known} remains reportable. Aggregate mean cost and cost per success are reported only when c_i is complete for every attempt. Cost per success is additionally undefined when there are no eligible successes. Costs from unsuccessful infrastructure or grader attempts remain in the numerator; missing costs are never set to zero.

The reported useful-work-per-dollar ratio is $\bar{u}_s/\bar{c}_s$, with the eligible-attempt denominator for mean utility and the all-attempt denominator for mean cost. It is descriptive utility per measured priced-resource dollar, not a monetary return. Expected elapsed time per success is $\sum_{i \in A_s} t_i / \sum_{i \in E_s} y_i$, including elapsed time for unsuccessful attempts. It is unknown unless elapsed time is known for every attempt and at least one eligible attempt succeeds. Median elapsed time describes individual episodes, not a successful retry sequence. Diagnostic means, medians, sample variances, and P10/P90 values are accompanied by known and unknown counts; unavailable values are not imputed.

A descriptive frontier retains only configurations with complete measured-scope priced-resource coverage, then identifies those not dominated in eligible success rate and mean measured priced-resource cost. Its relative productive-efficiency score is

$$\eta_s = \frac{\min_{j: \hat{p}_j \geq \hat{p}_s} \bar{c}_j}{\bar{c}_s}, \quad (3)$$

when all required comparisons are covered and the denominator is positive. This score depends on the tested configurations and the frozen prices. It is not a bound on achievable efficiency. Failed-attempt spend and retry counts are diagnostics; they do not establish avoidable or causal waste. A confidence interval that includes zero is an unresolved difference, not evidence of equivalence.

5 Experimental protocol and statistical analysis

The frozen plan is a full campaign with five repeats, three configured routes, and 30 task IDs. A task ID is assigned to one of 11 family-template clusters. Repeats and variants within a cluster are related. The assembler requires 45 immutable terminal run reports and exact selection and sample bindings before it can produce a full release input. A partial report is accepted only when its completed and failed attempts account for every planned task, with no cancellation or budget skip. Failed attempts are retained; they are not replaced with new attempts. The analyzer validates the campaign-frozen analysis protocol, system fingerprints, task cluster mapping, price vectors, observation identities, and resource events before calculating results.

The primary success threshold is 1.0. The configured sampling unit for uncertainty is the task-template cluster. Cost and ratio intervals use paired template-cluster resampling that keeps variants and repeats within their template and preserves shared task allocations across systems. With only 11 clusters, such intervals are descriptive and may be unstable. Repeated observations do not create additional independent templates. Any binomial interval should be labeled descriptive because it does not account for template dependence. Template-level results and a leave-one-template-out sensitivity display accompany the outcome summaries. The observed frontier reports measured quality and cost jointly. It does not establish an overall winner: success-rate differences remain unresolved, and the priced-resource subset excludes other operating costs.

5.1 Execution recovery and failure classification

The campaign producer maintains scoped worker leases and admits only the frozen run selections. The earlier diagnostic campaign prompted changes to terminal checkpoints, complete partial reports, and lease renewal during blocking API polling. During the new campaign, admission paused

because the local ledger still had thirty million micro-USD of development credits while the frozen combined exposure guard allowed fifty million. Native development-credit accounting aligned that test balance with the guard. A rejected admission was reconciled against its quote and deterministic idempotency key before retry. Resume handling was also amended to reuse the original immutable batch export after verifying all evidence fields, allowing only the exporter-generated timestamp to differ. A later supervisor correction isolated missed lease heartbeats to the captured worker process and lease generation, rather than stopping every worker in the batch. Shutdown allows bounded sandbox reads and terminal metering to finish before closing process handles, then releases interrupted leases through the native owner and generation fence. Existing checkpoints and dispatched-operation receipts are retained. These changes preserve task inputs, graders, model routes, runtime images, prices, and selected attempts. Expired units use native queue lease and operation fences. Recovery and producer revisions are recorded separately from the frozen campaign intent.

Producer provenance. The frozen campaign intent is `e64755463aaa4111632f5cbe5e6c06692086bda22bf4bd3f275c53a82ccd0d2f`. It identifies controller revision `a72126b33303528543818ba4df0429242d07898c` with producer-file SHA-256 `cc1cd88b9edf13180dbc2efcce8f140af5bc9480a5b9b349f9be1bd3e6f1a942`. The recovery record separately identifies immutable-export resume revision `2d59282ecef4a61ecd38ce283168fcf50c80b345` with producer-file SHA-256 `43527dbd621c4f5ac11ca071dbc579697a1e42f76fc5166514355191cbf754a0`, and fenced worker recovery revision `2a6cf34b34b20f36117408b26410809611975225` with producer-file SHA-256 `d94597aa939bd4669def09035046a46b9a53b457e57d7d1f5777df56ab0005fb`. These controller identities are separate from the frozen task packages and guest images. They do not establish an exact revision boundary for every attempt or identical execution conditions across the campaign. The comparison retains every selected attempt and supports descriptive observations rather than an isolated causal model or recovery effect.

An earlier 450-attempt campaign served as a measurement diagnostic. It exposed two problems in terminal sandbox accounting: the meter rejected valid all-zero counter responses and accepted positive responses before the provider counters settled. Its unavailable measurements cannot be reconstructed by imputing zeros. That campaign is retained separately and is not pooled with the new campaign or used to replace its failures.

The revised meter records protocol `evalrouter.costbench.modal-resource-usage.v2`. It samples terminal counters after resource shutdown and accepts them only when two consecutive valid responses agree and both are at least ten seconds after terminal observation. Samples are bounded to a twenty-second collection window. A stable tuple of nonnegative zeros is a valid provider accounting response. Invalid responses and collection errors retain closed diagnostic categories; unresolved counters remain unavailable. Execution and grading resources are stopped concurrently so collection for one resource does not keep the other running. Bounded qualification probes established that some positive counters changed after shutdown and that immediate workloads could have stable zero counters. These checks validate the collection procedure, rather than independent process measurements or provider invoices.

Analysis reads a sanitized, versioned JSON input and a frozen price vector; it does not call models or change prices. The assembler binds reports and receipts into that schema but does not synthesize missing quantities. Tables and charts must share the digest of the same input. Analysis reproduction is therefore distinct from repeating model execution, which may be stochastic and may require paid routes and credentials.

6 Results

The release reconciles all 450 selected attempts to 45 immutable terminal reports. The source and derived-result digests are 98e6afe2787a8e498383d7d2fac00a38198c6b000bc4987a2ce3725785c13772 and ceb4456b43fae35bdea010930f6d5d7f9a1f8c364ba616ba1f264612b411ed86. Integrity checking and scoped export provenance are recorded separately; offline checksums do not independently authenticate the provider or source operator.

6.1 Outcomes and uncertainty

GPT-4.1 achieves 146/150 successful attempts (97.3%), GPT-4.1 mini 132/150 (88.0%), and Haiku 140/150 (93.3%). Every selected attempt remains eligible. All ten infrastructure or execution failures occur in the Haiku cohort and count as unsuccessful outcomes. Partial grades contribute to mean utility without meeting the success threshold.

Table 2: Aggregate task outcomes. Confidence intervals use paired template-cluster bootstrap except where marked unavailable.

Configuration	Primary / Success rate excluded	Success rate CI	Mean utility CI	Median wall s	Expected time / success s
GPT-4.1	150 / 0	0.973 [0.937, 1.000]	0.990 [0.974, 1.000]	18.8855	25.3904
GPT-4.1 mini	150 / 0	0.880 [0.714, 1.000]	0.908 [0.746, 1.000]	22.9768	28.5579
Haiku 4.5	150 / 0	0.933 [0.813, 1.000]	0.933 [0.813, 1.000]	31.0724	51.4027

Table 3: Paired template-cluster percentile intervals for differences (system B minus system A).

System A	System B	Interpretation	Utility difference CI	Success difference CI	Cost/success difference CI
GPT-4.1	GPT-4.1 mini	difference_ observed	[-0.231, 0.000]	[-0.237, 0.000]	[-0.007, -0.001]
GPT-4.1	Haiku 4.5	difference_ observed	[-0.181, 0.017]	[-0.179, 0.041]	[0.003, 0.007]
GPT-4.1 mini	Haiku 4.5	difference_ observed	[-0.150, 0.226]	[-0.125, 0.257]	[0.004, 0.012]

Table 4: Reliability denominator under the all-selected-attempt protocol. Every selected attempt remains in the primary denominator.

Configuration	Success / Primary rate primary	Primary rate	Success / all attempts	All-attempt rate	Excluded attempts
GPT-4.1	146 / 150	0.973	146 / 150	0.973	0
GPT-4.1 mini	132 / 150	0.880	132 / 150	0.880	0
Haiku 4.5	140 / 150	0.933	140 / 150	0.933	0

All three paired success-rate intervals include zero, including an endpoint of zero for GPT-4.1 versus mini. These descriptive, unadjusted intervals do not establish either a population ranking or equivalence. The outcome sensitivity, family summaries, and leave-one-template-out results must be read with the small number of shared templates.

6.2 Cost coverage and execution time

The declared priced-resource scope is complete for all 450 attempts. It comprises provider-reported inference-token usage and sandbox resource counters, repriced under the frozen published-rate vector. Mean cost per attempt is \$0.005335 for GPT-4.1, \$0.002026 for mini, and \$0.009653 for Haiku. Dividing each configuration’s total measured cost by its successes gives \$0.005481, \$0.002303, and \$0.010342 per success. These values include measured cost on unsuccessful attempts. Mini and GPT-4.1 form the observed quality-cost frontier; Haiku has lower observed quality and higher measured cost than GPT-4.1.

All 890 execution and grading sandbox resources have recorded terminal usage: 450 environment resources and 440 grader resources. The original provider reports, scoped resource records, and audited analysis bindings retain separate provenance. Seven DNS failures and two connection failures have exact source-bound evidence that no provider request began. Their nine receipts establish zero inference use without imputing missing dispatched usage. A separately frozen compatibility-route contract establishes unsupported prompt caching for the Haiku route. Authoritative aggregate completion usage is priced once; Haiku’s separate reasoning-output count remains unavailable in all 150 attempts, but does not leave output pricing unresolved. GPU counters are zero for these CPU-only sandboxes and do not measure managed-inference hardware.

The independent accounting reconciliation covers 45 reports and 450 attempts, with no duplicate inference receipts or unassigned charges. Original quote recomputation and ledger arithmetic pass. All 15 reports per configuration match their frozen quote prices within 1%, with zero difference at the reported micro-USD precision. This validates configured-price accounting, not provider invoices or customer revenue. Analysis-vector repricing remains a separate basis from those quotes and local synthetic-credit debits.

Table 5: Measured priced-resource subset cost. Incomplete configurations have no comparative cost estimate.

Configuration	Mean subset (USD)	Subset / success CI Coverage
GPT-4.1	0.0053	0.005 [0.003, 0.009] Complete within scope
GPT-4.1 mini	0.0020	0.002 [0.002, 0.003] Complete within scope
Haiku 4.5	0.0097	0.010 [0.006, 0.015] Complete within scope

Episode elapsed time is available for all attempts. Including elapsed time from every attempt and dividing by successes yields 25.39 seconds per success for GPT-4.1, 28.56 seconds for mini, and 51.40 seconds for Haiku. Episode completion is recorded before terminal metering, so these intervals exclude subsequent counter collection and do not establish worker throughput. Median gateway intervals are 945, 984, and 1,666 milliseconds, respectively; median provider dispatch-to-response intervals are 898, 913, and 1,600 milliseconds. Reset, tool, and grade operation intervals

Table 6: Resource-price coverage within the declared measured scope. Missing reasoning splits are diagnostic when authoritative aggregate completion usage is priced once. This table does not assert total economic-cost coverage.

Configuration	Scope	Unresolved priced quantities	Unpriced resources	Known subtotal
GPT-4.1	Complete	None recorded	None recorded	0.8002
GPT-4.1 mini	Complete	None recorded	None recorded	0.3039
Haiku 4.5	Complete	None recorded	None recorded	1.4479

are recorded separately. These buckets can overlap. Subtracting them does not establish isolated platform or harness overhead, and elapsed time is not independently priced.

System comparison

System	Success (95% interval)	Mean utility (95% interval)	Mean measured subset (USD)	Known subtotal (USD)	Subset / success (USD)	Observed efficiency
gpt-4.1	0.9733 [0.9375, 1]	0.99 [0.9741, 1]	0.005335	0.8002	0.005481 [0.002762, 0.008201]	1
gpt-4.1-mini	0.88 [0.7143, 1]	0.9078 [0.7459, 1]	0.002026	0.3039	0.002303 [0.001522, 0.003084]	1
claude-haiku-4.5	0.9333 [0.8129, 1]	0.9333 [0.8129, 1]	0.009653	1.448	0.01034 [0.006291, 0.01469]	0.5527

costbench-v1 CostBench@costbench-v1 · formula costbench-v2-all-attempts · prices costbench-published-2026-10-02-v1 basis
dated 2026-10-02 · system prices gpt-4.1=costbench-gpt-4.1-published-2026-10-02-v1.
gpt-4.1-mini=costbench-gpt-4.1-mini-published-2026-10-02-v1.
claude-haiku-4.5=costbench-claude-haiku-4.5-published-2026-10-02-v1 · source
99e6afe2787a8e498383d7d2fac00a38198c6b000bc4987a2ce3725785c13772 · 2026-10-03 · systems: gpt-4.1, gpt-4.1-mini,
claude-haiku-4.5

Figure 2: Configured-system outcomes and measured priced-resource costs. Intervals use paired template-cluster bootstrap; the priced scope excludes other operating costs.

7 Discussion and limitations

The protocol is intended to make the evidence behind an agent comparison inspectable. It identifies a full configured system, grades task outcomes from tests or final state, preserves resource receipt provenance, and applies explicit prices in a separate analysis. The campaign records task outcomes and complete costs within the declared measured priced-resource subset. Keeping that subset separate from invoices and unmeasured operating costs prevents a broader economic claim than the evidence supports. The observed outcome percentages do not establish population-level differences under the declared uncertainty analysis.

The workload is synthetic and small. Thirty task IDs share only 11 templates, and the three families represent selected software repair, database, and API state-change operations. The benchmark does not estimate broad production task distributions, customer value, or universal model capability. Hidden task criteria can reduce direct answer leakage but cannot demonstrate absence of training contamination. Results describe the specific taskset, systems, tools, harnesses, runtime, budgets, and price versions in the released manifest.

The recorded campaign includes execution interruptions and amendments to the local supervisor’s recovery handling. Task inputs, graders, model routes, guest runtime images, resource limits,

and the selected attempt population were retained. These operational changes do not establish identical execution conditions throughout the campaign. All recorded errors remain in the primary denominator and all recorded resource use remains in the priced scope. The comparison describes the observed campaign; neither its outcome differences nor its uncertainty intervals isolate a causal effect of model identity or of the recovery changes.

Resource coverage has practical limits. Provider-reported usage can be incomplete; provider route estimates are not invoices; sandbox resource telemetry is not billing reconciliation; and local synthetic-credit ledger debits do not establish collected revenue or customer payment. No independent invoice reconciliation is claimed. Physical resource prices do not establish actual marginal infrastructure cost. Latency measures can overlap and may omit reset, grading, or orchestration time. Any unmeasured or unpriced resource constrains an economic conclusion. Operation labels such as retry or duplicate request are descriptive and do not establish causal waste or avoidable work.

Finally, model and harness effects are intentionally combined in the unit of comparison. Differences cannot be attributed to model identity alone. Small-cluster uncertainty further limits ranking. These limits must remain adjacent to the results they qualify, including plots and any derived selection guidance.

8 Reproducibility

The release capsule contains the sanitized 450-attempt analysis input, complete system configurations and fingerprints, opaque task IDs and template membership, frozen prices and source digests, analyzer code, generated results and figures, manuscript source, and rendered PDF. Its manifests bind source, derived data, code, dependencies, and paper artifacts by checksum. Scored task values, private seeds, hidden grading criteria, raw model responses, credentials, and private endpoints are excluded. The internal workspace identifier is replaced by a stable scope digest in the public provenance projection.

The scored workload uses execution source revision `05eab58d744cab13fede33aaa8104f166b65485f`. The scored workload implementation and protected task criteria remain private. The public capsule supports offline reproduction of the released analysis, rather than independent reconstruction of hidden task inputs or authentication of private reports. A separate one-task API example demonstrates the execution interface and is excluded from the 450-attempt cohort. Its execution qualification does not establish hosted public availability.

From an extracted capsule with Python 3.12 and the declared plotting dependency installed, the offline command is:

```
PYTHONPATH=analysis python3.12 analysis/analyze.py
analysis/source_evidence.json -output-dir reproduction
-mode release -success-threshold 1.0
-score-direction higher_is_better -seed 20261002
-bootstrap-replicates 2000
```

The capsule README provides the copyable command and checksum verification instructions. Offline analysis performs no model calls. It reproduces derived statistics from frozen evidence; repeating paid model execution can produce different outcomes.

9 Conclusion

Kimpton CostBench records 450 attempts on a small synthetic operational workload and links outcomes to resource evidence and frozen prices. All selected attempts and all 890 sandbox resource measurements are accounted for. GPT-4.1 and mini form the observed quality-cost frontier, while shared-template uncertainty does not resolve success-rate differences. The released record supports measured priced-resource comparisons and offline analysis reproduction. It does not establish total operating cost, invoice reconciliation, identical operational conditions, or a universal model ranking.

A Task template inventory

Table 7 lists the implementation-level task kinds for documentation. It omits task input values, hidden tests, expected rows, and expected states. Public evidence identifies task variants by opaque IDs and template membership.

Table 7: Template inventory without task answers.		
Family	Templates	Operation class
Software repair	Invoice rounding; inventory reservation; retry idempotency	Repair a pure function, then grade by isolated functional cases.
Database	Insert reconciliation; inventory update; ledger report; migration backfill	Mutate isolated SQLite state through bounded SQL and grade captured final rows.
Simulated API	Deployment rollout; ticket reconciliation; access policy; quota update	Change local simulated service state and grade captured resources.

B System identity and controls

The release source records the full configuration, rather than only a model label. All configurations use the native `evalrouter.costbench.v1` harness, temperature zero, no requested provider seed, a maximum of 20 steps, three provider-generation attempts, and receipt-only tool-operation recovery. Family-specific tool schemas, policy hashes, runtime contract digests, budgets, context policies, and family fingerprints are included in the sanitized systems document. Its declared context policy allows 2,048 output tokens; the native provider route applies a separate 1,024-token request cap. The system runtime-image field is a canonical digest over family image references, while each resource receipt records its concrete image digest. These distinct fields are not interchangeable. The per-run cap is four million micro-USD in the local synthetic-credit ledger. A fifty-million micro-USD exposure guard covers the earlier diagnostic campaign and the new measurement campaign together. These controls constrain test-ledger accounting and do not establish provider invoices or cash payment.

Table 8: Parent configuration identities. Family execution fingerprints and complete configuration objects are supplied in the release source.

Label	Configuration fingerprint
GPT-4.1	75c70d500b04cc303798a999372eee9aebc0a62f1dbf1bd241249cf9125f7f5c
GPT-4.1 mini	bf8ac685eedb94fae5ba2690aefb5b890afed0653835746ce6672b4e89995ffd
Claude Haiku 4.5	d0958e964f8b06f9704cfb82c261f69ce33781ab0d174300e40deff7451ab4c5

C Receipt and resource coverage table

Table 9: Descriptive per-attempt latency and operation counts. Coverage is shown for every measure.

Configuration	Measure	Known / unknown	Mean	Median	Variance	P10 / P90
GPT-4.1	episode_ elapsed_ seconds	150 / 0	24.71	18.89	149.75	14.68, 42.47
GPT-4.1	model_calls	150 / 0	3.39	2.00	5.50	2.00, 7.00
GPT-4.1	tool_calls	150 / 0	2.47	1.00	5.97	1.00, 6.00
GPT-4.1	db_calls	150 / 0	0.53	0.00	1.23	0.00, 1.00
GPT-4.1	api_calls	150 / 0	0.33	0.00	0.22	0.00, 1.00
GPT-4.1	retries	150 / 0	0.01	0.00	0.01	0.00, 0.00
GPT-4.1 mini	episode_ elapsed_ seconds	150 / 0	25.13	22.98	95.66	14.82, 37.74
GPT-4.1 mini	model_calls	150 / 0	3.39	2.00	3.25	2.00, 6.00
GPT-4.1 mini	tool_calls	150 / 0	2.43	1.00	3.71	1.00, 5.00
GPT-4.1 mini	db_calls	150 / 0	0.99	0.00	4.49	0.00, 4.00
GPT-4.1 mini	api_calls	150 / 0	0.33	0.00	0.22	0.00, 1.00
GPT-4.1 mini	retries	150 / 0	0.04	0.00	0.04	0.00, 0.00
Haiku 4.5	episode_ elapsed_ seconds	150 / 0	47.98	31.07	13704.99	17.76, 57.42
Haiku 4.5	model_calls	150 / 0	3.99	4.00	3.64	2.00, 7.00
Haiku 4.5	tool_calls	141 / 9	3.40	3.00	5.60	1.00, 7.00
Haiku 4.5	db_calls	150 / 0	1.19	0.00	3.35	0.00, 3.00
Haiku 4.5	api_calls	150 / 0	0.33	0.00	0.22	0.00, 1.00
Haiku 4.5	retries	150 / 0	0.01	0.00	0.01	0.00, 0.00

D Failure handling and uncertainty

The generated release appendix reports normalized observation-status counts, the closed derived failure class, eligibility, known priced subtotals, and complete measured-scope costs when coverage permits. Family summaries, template counts, leave-one-template-out sensitivity, and price and billing bases are also generated from the same validated result. Bootstrap seed and replicate count are recorded in generated metadata. Every generated table is bound to the reconciled source and

result digests.

Table 10: Counts by normalized observation status and derived failure class. Every selected attempt is in the primary population.

Configuration	Normalized status	Derived class	Outcome	Attempts
Haiku 4.5	benchmark_ infrastructure_ failure	benchmark_ infrastructure_failure	Primary	8
Haiku 4.5	completed	None reported	Primary	140
Haiku 4.5	failed	execution	Primary	2
GPT-4.1	completed	None reported	Primary	150
GPT-4.1 mini	completed	None reported	Primary	150

Table 11: Resource accounting by derived failure class. Known subtotal sums available priced components; full cost is shown only when every attempt in that row is complete within scope.

Configuration	Derived class	Outcome	Attempts	Known subtotal (USD)	Complete cost (USD)	Incomplete
Haiku 4.5	None reported	Primary	140	1.3666	1.3666	0
Haiku 4.5	benchmark_ infrastructure_ failure	Primary	8	0.0609	0.0609	0
Haiku 4.5	execution	Primary	2	0.0204	0.0204	0
GPT-4.1	None reported	Primary	150	0.8002	0.8002	0
GPT-4.1 mini	None reported	Primary	150	0.3039	0.3039	0

Table 12: Frozen task-family and template counts.

Family	Tasks	Templates
swe	10	3
database	10	4
api	10	4

Table 13: Family outcomes and sample utility variance. Each family has only three or four task-template clusters; these aggregates do not have cluster-bootstrap intervals.

Family	Configuration	Primary / excluded	Success rate	Mean utility	Utility variance
swe	GPT-4.1	50 / 0	0.9400	0.9900	0.0016
swe	GPT-4.1 mini	50 / 0	0.9000	0.9833	0.0026
swe	Haiku 4.5	50 / 0	0.8000	0.8000	0.1633
database	GPT-4.1	50 / 0	0.9800	0.9800	0.0200
database	GPT-4.1 mini	50 / 0	0.7400	0.7400	0.1963
database	Haiku 4.5	50 / 0	1.0000	1.0000	0.0000
api	GPT-4.1	50 / 0	1.0000	1.0000	0.0000
api	GPT-4.1 mini	50 / 0	1.0000	1.0000	0.0000
api	Haiku 4.5	50 / 0	1.0000	1.0000	0.0000

Table 14: Family cost and latency summaries. Cost fields remain unknown when family attempt coverage is incomplete.

Family	Configuration	Mean subset cost	Cost variance	Cost / success	Expected time / success s
swe	GPT-4.1	0.0104	0.0000	0.0110	38.9600
swe	GPT-4.1 mini	0.0027	0.0000	0.0030	33.4394
swe	Haiku 4.5	0.0165	0.0000	0.0206	115.4322
database	GPT-4.1	0.0033	0.0000	0.0034	20.4078
database	GPT-4.1 mini	0.0022	0.0000	0.0029	36.8098
database	Haiku 4.5	0.0084	0.0000	0.0084	31.3320
api	GPT-4.1	0.0024	0.0000	0.0024	17.5180
api	GPT-4.1 mini	0.0012	0.0000	0.0012	18.0581
api	Haiku 4.5	0.0040	0.0000	0.0040	20.2498

Table 15: Leave-one-template-out descriptive sensitivity. Each row removes one shared template and retains all variants and repeats in the other templates. This is not an independent resampling analysis.

Removed template	Configuration	Primary attempts	Mean utility	Success rate	Cost / success
api:access_policy	GPT-4.1	140	0.9893	0.9714	0.0057
api:access_policy	GPT-4.1 mini	140	0.9012	0.8714	0.0024
api:access_policy	Haiku 4.5	140	0.9286	0.9286	0.0108
api:deployment_rollout	GPT-4.1	135	0.9889	0.9704	0.0058
api:deployment_rollout	GPT-4.1 mini	135	0.8975	0.8667	0.0025
api:deployment_rollout	Haiku 4.5	135	0.9259	0.9259	0.0111
api:quota_update	GPT-4.1	140	0.9893	0.9714	0.0057
api:quota_update	GPT-4.1 mini	140	0.9012	0.8714	0.0024
api:quota_update	Haiku 4.5	140	0.9286	0.9286	0.0108
api:ticket_reconciliation	GPT-4.1	135	0.9889	0.9704	0.0058
api:ticket_reconciliation	GPT-4.1 mini	135	0.8975	0.8667	0.0024
api:ticket_reconciliation	Haiku 4.5	135	0.9259	0.9259	0.0111
database:insert_reconcile	GPT-4.1	135	0.9889	0.9704	0.0058
database:insert_reconcile	GPT-4.1 mini	135	0.8975	0.8667	0.0024
database:insert_reconcile	Haiku 4.5	135	0.9259	0.9259	0.0107
database:inventory_update	GPT-4.1	135	0.9889	0.9704	0.0058
database:inventory_update	GPT-4.1 mini	135	0.8975	0.8667	0.0024
database:inventory_update	Haiku 4.5	135	0.9259	0.9259	0.0107
database:ledger_report	GPT-4.1	140	0.9964	0.9786	0.0054
database:ledger_report	GPT-4.1 mini	140	0.9726	0.9429	0.0020
database:ledger_report	Haiku 4.5	140	0.9286	0.9286	0.0102
database:migration_backfill	GPT-4.1	140	0.9893	0.9714	0.0057
database:migration_backfill	GPT-4.1 mini	140	0.9226	0.8929	0.0021
database:migration_backfill	Haiku 4.5	140	0.9286	0.9286	0.0105
swe:inventory_reservation	GPT-4.1	135	0.9889	0.9704	0.0054

Removed template	Configuration	Primary attempts	Mean utility	Success rate	Cost / success
swe:inventory__reservation	GPT-4.1 mini	135	0.8975	0.8667	0.0023
swe:inventory__reservation	Haiku 4.5	135	0.9926	0.9926	0.0097
swe:invoice__rounding	GPT-4.1	130	0.9923	0.9923	0.0036
swe:invoice__rounding	GPT-4.1 mini	130	0.9000	0.9000	0.0021
swe:invoice__rounding	Haiku 4.5	130	0.9308	0.9308	0.0081
swe:retry__idempotency	GPT-4.1	135	0.9889	0.9704	0.0055
swe:retry__idempotency	GPT-4.1 mini	135	0.8975	0.8667	0.0022
swe:retry__idempotency	Haiku 4.5	135	0.9259	0.9259	0.0099

Table 16: Frozen per-system price-vector rates, scaled to readable units.

Configuration	Vector	Basis date	Resource and unit	Scaled rate
All configurations	costbench-published-2026-10-02-v1	2026-10-02	cpu_core_nanosecs (CPU core-seconds)	3.942e-05 USD / CPU core-seconds
All configurations	costbench-published-2026-10-02-v1	2026-10-02	mem_gib_nanosecs (GiB-seconds)	6.67e-06 USD / GiB-seconds
GPT-4.1	costbench-gpt-4.1-published-2026-10-02-v1	2026-10-02	cached_input_tokens (million tokens)	0.5 USD / million tokens
GPT-4.1	costbench-gpt-4.1-published-2026-10-02-v1	2026-10-02	input_tokens (million tokens)	2 USD / million tokens
GPT-4.1	costbench-gpt-4.1-published-2026-10-02-v1	2026-10-02	output_tokens (million tokens)	8 USD / million tokens
GPT-4.1	costbench-gpt-4.1-published-2026-10-02-v1	2026-10-02	reasoning_output_tokens (million tokens)	8 USD / million tokens
GPT-4.1	costbench-gpt-4.1-published-2026-10-02-v1	2026-10-02	total_output_tokens (million tokens)	8 USD / million tokens
GPT-4.1 mini	costbench-gpt-4.1-mini-published-2026-10-02-v1	2026-10-02	cached_input_tokens (million tokens)	0.1 USD / million tokens
GPT-4.1 mini	costbench-gpt-4.1-mini-published-2026-10-02-v1	2026-10-02	input_tokens (million tokens)	0.4 USD / million tokens

Configuration	Vector	Basis date	Resource and unit	Scaled rate
GPT-4.1 mini	costbench-gpt-4.1-mini-published-2026-10-02-v1	2026-10-02	output_tokens (million tokens)	1.6 USD / million tokens
GPT-4.1 mini	costbench-gpt-4.1-mini-published-2026-10-02-v1	2026-10-02	reasoning_output_tokens (million tokens)	1.6 USD / million tokens
GPT-4.1 mini	costbench-gpt-4.1-mini-published-2026-10-02-v1	2026-10-02	total_output_tokens (million tokens)	1.6 USD / million tokens
Haiku 4.5	costbench-claude-haiku-4.5-published-2026-10-02-v1	2026-10-02	cached_input_tokens (million tokens)	0.1 USD / million tokens
Haiku 4.5	costbench-claude-haiku-4.5-published-2026-10-02-v1	2026-10-02	input_tokens (million tokens)	1 USD / million tokens
Haiku 4.5	costbench-claude-haiku-4.5-published-2026-10-02-v1	2026-10-02	output_tokens (million tokens)	5 USD / million tokens
Haiku 4.5	costbench-claude-haiku-4.5-published-2026-10-02-v1	2026-10-02	reasoning_output_tokens (million tokens)	5 USD / million tokens
Haiku 4.5	costbench-claude-haiku-4.5-published-2026-10-02-v1	2026-10-02	total_output_tokens (million tokens)	5 USD / million tokens

All configurations: costbench-published-2026-10-02-v1, basis dated 2026-10-02; source modal official published pricing, <https://modal.com/pricing>; SHA-256 5c7dbb6f204fc42f03bcab525e78e825eb04005cd8f6cc11f4a00d12f93c0cce.

Haiku 4.5: costbench-claude-haiku-4.5-published-2026-10-02-v1, basis dated 2026-10-02; source claude-haiku-4.5 official published pricing, <https://platform.claude.com/docs/en/about-claude/pricing>; SHA-256 8b3f4e29e1c79ba465e242b1f3dc7c66d6d789c7f5f1195477ae8a9e09083bb4.

GPT-4.1: costbench-gpt-4.1-published-2026-10-02-v1, basis dated 2026-10-02; source gpt-4.1 official published pricing, <https://developers.openai.com/api/docs/models/gpt-4.1>; SHA-256 3e1014be76723c4da7ac4fbf46aed950f2b2b3c12ee83e49598b5f845eb32190.

GPT-4.1 mini: costbench-gpt-4.1-mini-published-2026-10-02-v1, basis dated 2026-10-02; source gpt-4.1-mini official published pricing, <https://developers.openai.com/api/docs/models/gpt-4.1-mini>; SHA-256 68c9d36d280ff051e38f4b60c9b2518e871589b3e139dfc17912f0cce379b22.

Table 17: Original configured-price and local test-ledger receipt gaps before the separate no-dispatch audits. Those nine proofs establish zero provider use for the audited calls; the ledger records and debits remain unchanged. Known values are not collected revenue or provider invoices, and remain separate from sandbox resource repricing.

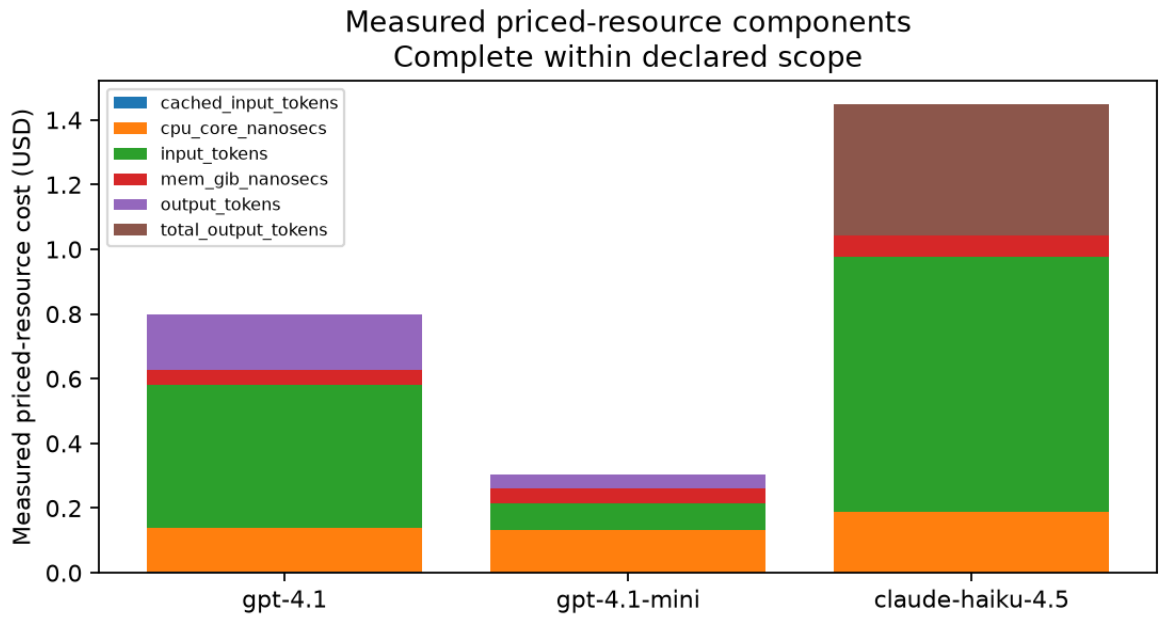
Configuration	Known configured token estimate subtotal (micro-USD)	Pre-audit provider receipt gaps	Known local test-ledger debit (micro-USD)	Pre-audit local-ledger receipt gaps
GPT-4.1	618588	1	618588	1
GPT-4.1 mini	126520	6	126520	6
Haiku 4.5	1195705	2	1195705	2

E Price vector and reproduction

The generated price tables record each vector’s version, currency, declared date, source URL and digest, and priced resource units. The published-page basis was observed on October 2, 2026; that observation does not establish a provider tariff change on that date. Historical route estimates remain separately labeled configured-price estimates. No estimate is called a provider invoice or cash payment.

F Recorded resource components

The figure below reports recorded inference and sandbox resource price components under the frozen vector. These components cover the declared priced-resource subset, rather than total operating cost. All seven principal release graphics are supplied in the capsule.



costbench-v1 CostBench@costbench-v1 · formula costbench-v2-all-attempts · prices costbench-published-2026-10-02-v1 basis
dated 2026-10-02 · system prices gpt-4.1=costbench-gpt-4.1-published-2026-10-02-v1,
gpt-4.1-mini=costbench-gpt-4.1-mini-published-2026-10-02-v1,
claude-haiku-4.5=costbench-claude-haiku-4.5-published-2026-10-02-v1 · source
98e6afe2787a8e498383d7d2fac00a38198c6b000bc4987a2ce3725785c13772 · 2026-10-03 · systems: gpt-4.1, gpt-4.1-mini,
claude-haiku-4.5

Figure 3: Recorded inference and sandbox resource quantities repriced under the frozen vector. Bars describe the declared measured priced-resource subset.

References

- [1] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? *arXiv preprint arXiv:2310.06770v3*, 2023. URL <https://arxiv.org/abs/2310.06770v3>.
- [2] Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter. *arXiv preprint arXiv:2407.01502v1*, 2024. URL <https://arxiv.org/abs/2407.01502v1>.
- [3] Percy Liang et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110v2*, 2022. URL <https://arxiv.org/abs/2211.09110v2>.
- [4] Jiayu Liu, Cheng Qian, Zhaochen Su, Qing Zong, Shijue Huang, Bingxiang He, and Yi R. Fung. CostBench: Evaluating multi-turn cost-optimal planning and adaptation in dynamic environments for LLM tool-use agents. *arXiv preprint arXiv:2511.02734v3*, June 2026. URL <https://arxiv.org/abs/2511.02734v3>. First submitted November 2025; version 3 revised 29 June 2026.
- [5] Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ -bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045v1*, 2024. URL <https://arxiv.org/abs/2406.12045v1>.