Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

# LEDGER: Deterministic Checks and Auditable Evaluation for Business AI Systems

**Ben Sirb**

ben@ai-interval.com

Interval

**Peter A. Devine**

Interval

**Nick Ryan**

Interval

**Alex Chiocchi**

Interval

**Research Article**

**Keywords:** LLM evaluation, business intelligence, deterministic checks, LLM-as-judge, cost per correct answer, open-weight models, auditable evaluation

**Posted Date:** November 9th, 2026

**DOI:** https://doi.org/10.21203/rs.3.rs-11639610/v1

**License:** This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

**Additional Declarations:** The authors declare potential competing interests as follows: All authors are employees of Interval, which builds the business intelligence assistant evaluated in this work and the LEDGER platform itself.

# LEDGER: Deterministic Checks and Auditable Evaluation for Business AI Systems

**Ben Sirb, Peter A. Devine, Nick Ryan, Alex Chiocchi**
Interval
{ben, peter.devine, nick.ryan, alex}@ai-interval.com

## Abstract

Public benchmarks rank models on public tasks, but those scores do not directly predict performance on a company's own work. Companies need a practical way to retest their AI systems after every change. We present LEDGER, an evaluation platform that turns a company's own data, questions, and rules into a repeatable test. Deterministic checks score what they can, and a separate LLM judge and a deterministic evidence check handle the rest, with human review of sampled answers. Every score traces to evidence, and cost per correct answer makes the tradeoff between accuracy and cost explicit. Because each failure points to a scenario, expected value, and rule, LEDGER also guides improvement: coding agents repair the system while sampled human review validates what counts as correct. Interval used it to build its business intelligence assistant around DeepSeek V4 Flash (DS4), a privately hosted open-weight model. In Interval's harness, DS4 passes 96 out of 97 held-out scenarios, while five proprietary models and the open-weight GLM-5.3 pass 94 to 97 given the same tools and definitions. Even under their generally cheaper schema-only prompt, they cost an estimated 3 to 54 times as much per correct answer as bare self-hosted DS4. LEDGER makes this tradeoff measurable, showing how a company can achieve frontier-level accuracy while choosing its own balance of cost and control.

## 1 Introduction

Companies now place AI assistants over their own operational data (Lei et al., 2025; Zhang et al., 2026; Vyas et al., 2026). For example, an engineering manager might ask how many pull requests were reviewed last week, or an analyst might ask for the median time to resolve incidents in a given repository. The system behind such an assistant is more than just a model: a prompt, tools that query the company's databases, rules about what it may not reveal, and orchestration that decides which tool to call and when. Any of these parts can change from week to week, and each change can impact whether the answer is correct.

Public benchmarks rank models on public tasks such as writing SQL for a given schema (Yu et al., 2018; Li et al., 2023; Lei et al., 2025), not on the company's own questions, data, and rules. The tools and interface around a model change its results as much as the model itself (Yang et al., 2024), and no single harness wins across tasks (Gupta et al., 2026), so a model ranking based on public benchmarks does not predict a deployed system's accuracy. Public test items also leak into training data (Sainz et al., 2023), so a high public score may reflect memorisation. A company needs its own test, over its own data, that it can rerun after every change with verdicts it can inspect.

Our first contribution is a platform for that test that we call LEDGER (Logged Evidence for Deterministic, Governed Evaluation and Review). A scenario records three things: a question or an interface action, the expected outcome, and the evidence that decides whether that outcome was reached. We demonstrate the platform on Interval's business intelligence assistant, which answers questions over an engineering and operations warehouse and drives a dashboard. The evidence is either a frozen SQL query over a frozen copy of the warehouse, a fixture, a policy rule, or the expected dashboard state (Section 3.2). Where the outcome can be computed from any of that evidence, a deterministic check scores the answer and returns the same verdict every time. Where an answer states the value in prose the check cannot parse, the scenario enters a logged LLM review constrained by a deterministic evidence check. Every scenario carries a content hash, every run records the model identity and prompt hash, and raw outputs are kept, so every score traces to a run and the same scenarios can be rerun against

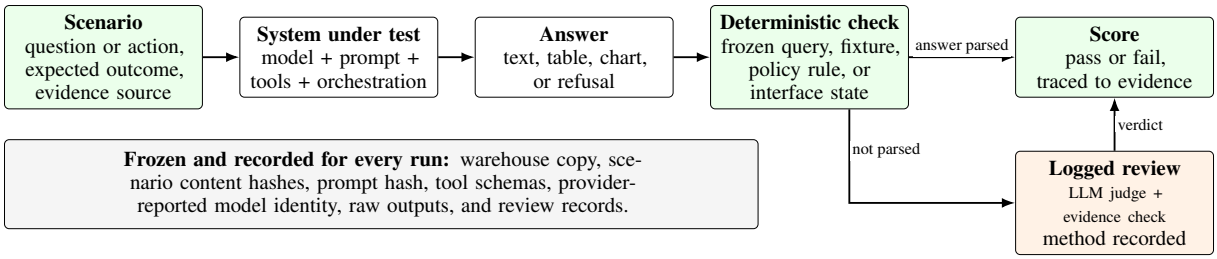



Figure 1: How LEDGER scores one scenario. The scenario and the check are fixed; the system under test can be replaced. An answer the check can parse is scored directly; one it cannot parse goes to an LLM judge constrained by a deterministic evidence check. The review method and score are recorded. Either way the score traces to evidence.

another system (Figure 1).

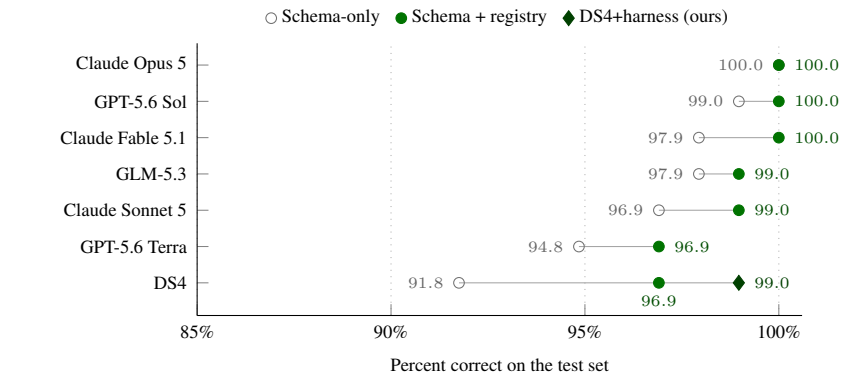
Our second contribution is a case study that uses the platform to measure how much of an assistant's accuracy comes from the model and how much from the system around it. Interval's deployed assistant runs DeepSeek V4 Flash (DeepSeek-AI, 2026), hereafter DS4, a privately hosted open-weight model, inside Interval's task harness (hereafter DS4+harness) containing the prompts, rule packs, skills, sub-agents, and four governed data tools around the model. We compare it with seven baselines: five proprietary models, the open-weight GLM-5.3 (together, the frontier models), and DS4 itself, each given the same four tools and a plain prompt, with and without the company's metric registry (terms are defined at the end of this section). On the test set, where scenarios, data, tools, and limits are the same for every system, bare DS4 against DS4+harness measures what the harness adds, and the baselines against each other measure what the model choice adds. The development-set results are supporting evidence only (Section 5.3).

The harness was built in repeated cycles. Each failing scenario, its expected value, and its rule pointed to a repair, the suite was rerun, and the score was recorded, under the standing rule that no repair may target a scenario. Coding agents drove those cycles from the recorded scores, and a human approved every scenario admission and every release, so the agents could change the system but not what counts as correct. The test set shows how the result holds on held-out content. LEDGER therefore did two jobs: it measured the system, and it was the instrument by which a privately hosted open-weight model reached frontier-level accuracy on this workload at a projected fraction of frontier cost (Figure 2).

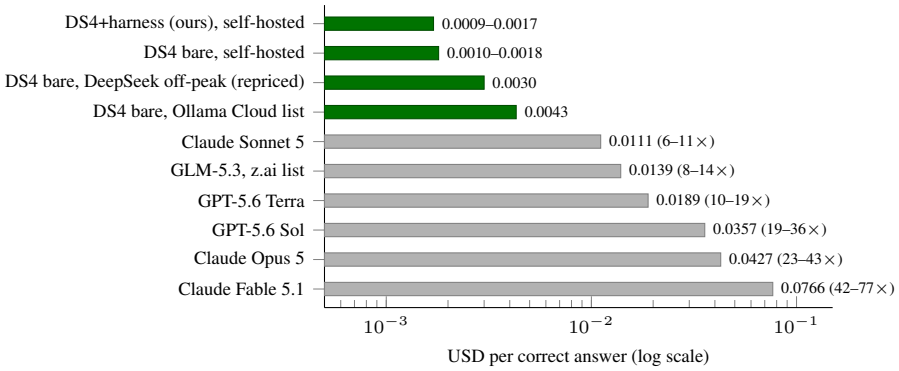
The measured result comes from 100 scenarios generated after the system was frozen, with values absent from the 968-scenario development set (Section 3.1), scored for every system with the same scoring method. We report the 97 whose keys survived a review of every question template. On it, DS4+harness and the six frontier models land within a few scenarios of each other, bare DS4 trails without the metric registry and closes most of that gap with it, and cost per correct answer differs by an estimated 6 to 77 times (Section 5).

If several models reach similar accuracy once a company's definitions are spelled out, the choice among them depends on cost and control. Making that choice safely requires a test that highlights which change helped on which scenarios, and at what cost per correct answer. LEDGER is built to be that test, which can be rerun on every change.

**Terms.** The *metric registry* is the company's catalog of metric definitions, column lists, and metric contracts, given to a model as prompt text. The *schema-only prompt* lists the warehouse tables and columns, and the *schema-plus-registry prompt* adds the registry. The *task harness* is the prompts, rule packs, skills, sub-agents, and governed data tools around DS4, and DS4+harness is DS4 inside it. An *oracle* is the stored deterministic check that decides a scenario. The *review judge* is the LLM that reads an answer the oracle cannot parse. Every judge pass must satisfy a deterministic *evidence check*, and an author audits sampled answers. Every review verdict is saved along with the method's identity and version. *Model-eligible* scenarios are those a model answers; clarification scenarios are answered by the server. *Held-out* scenarios carry new values on the development templates.




(a) Accuracy on the test set




(b) Cost per correct answer on the test set

Figure 2: The headline results, on the 97 test scenarios kept after a review of every question template. (a) Percent correct for six frontier models under two prompts (schema-only lists the warehouse tables and columns; schema-plus-registry adds the metric registry). DS4 is shown as one series of three points: bare with the schema-only prompt, bare with the registry, and inside Interval's task harness (DS4+harness). Opus scores the same under both prompts, so its two markers coincide. (b) Cost per correct answer under the schema-plus-registry prompt, with each frontier bar's multiple of the self-hosted bare DS4 range in brackets. API cost from recorded usage at provider rates for the five proprietary models, GLM-5.3 at its z.ai list price, and DS4 repriced at the Ollama Cloud list price for the tested checkpoint, repriced at DeepSeek's off-peak tariff, or self-hosted on Interval's hardware (an estimate that assumes continuous utilisation; bars show the upper end of the range in Table 4).

## 2 Related work

**Evaluating agents on data tasks.** Text-to-SQL benchmarks (Yu et al., 2018; Li et al., 2023; Lei et al., 2025) measure whether a model can write a correct query from a schema. Our test differs in two ways: the unit under test is the whole system rather than the model, and the check is the result of a frozen query over a frozen warehouse rather than a comparison between two queries. $\tau$-bench (Yao et al., 2024) scores tool-using agents against domain rules and a simulated user, which is the closest public analogue to our policy rules and multi-turn scenarios. DI-Bench (Zhang et al., 2026) generates enterprise data benchmarks in which business rules modify the computation, which is what our metric registry does. SWE-agent (Yang et al., 2024) showed that the interface an agent is given changes its results as much as the model does, and Gupta et al. (2026) find that no harness is superior across tasks and argue for harness-level evaluation. Belcak et al. (2025) argue that smaller models suit agentic workloads, and our case study measures a related position on one company's tasks.

**Evaluation frameworks.** Inspect (UK AI Security Institute, 2024) is a general framework for LLM evaluations, with custom scorers, structured evaluation logs, and tool and computer use. LEDGER's current standalone implementation runs on Inspect as its execution and logging layer, but the measurements in this paper come from the original implementation inside the product repository. What LEDGER adds is a company-specific workflow around the execution layer: answer keys computed from a frozen copy of the company's own warehouse, a content hash on every scenario that makes benchmark drift measur-

able (Section 5.3), review records that identify the judge and scoring-method version with a deterministic evidence check, and interface-state scenarios that check recorded dashboard transitions, exercised here on the development set only.

**LLM judges and evaluation statistics.** Zheng et al. (2023) established LLM judging and documented its biases, and Thakur et al. (2024) measured judge alignment and vulnerabilities. We use an LLM for the review step, which a deterministic evidence check constrains but which decides most scenarios. It is kept out of every system under test, and we audit samples of its verdicts. Miller (2024) argues for error bars and paired comparisons on shared items. Our scenarios are shared across all systems, and we report the judge's own rerun variability on the test set. Yuan et al. (2025) show that inference numerics alone change model outputs, which is why single runs are a stated limitation here. Sainz et al. (2023) describe benchmark contamination; its analogue here is development exposure, so the development-set figures are supporting evidence only (Section 5.3).

**Provenance.** Every LEDGER scenario names the query or fixture that decides it, and every review record identifies the judge and scoring-method version, so a number in this paper can be traced to a row; runtime tracing through the OpenTelemetry generative-AI conventions (OpenTelemetry Authors, 2026; Dong et al., 2024; Wang et al., 2026) is not part of the measurements here.

## 3 The LEDGER evaluation platform

Prior work shows that the interface and harness affect agent performance, with effects that vary across tasks (Yang et al., 2024; Gupta et al., 2026). A score alone therefore cannot tell a company whether a change in the model, prompt, tools, or scoring method caused the difference. LEDGER records the scenario, the evidence defining a correct outcome, the system configuration, and the scoring method separately so reviewers can identify what changed. The scenario and evidence stay fixed within a comparison. The system can be replaced, and the scoring records remain available for inspection.

### 3.1 Task definition

The assistant answers questions about an engineering and operations warehouse and drives a web dashboard of charts, tables, and data lineage. The 968 development scenarios were generated from question templates that an AI coding agent drafted under human curation, filled with real repository names, weeks, and metric values from that warehouse. Each template family carries one of seven need classes (Table 1). The agent wrote the questions, not the answers: every expected value is computed by a frozen query over the warehouse, read from a hand-entered fixture or fixed by a policy contract, and no model produced a gold value. Table 1 states what each class asks and what decides it; the multi-turn class tests whether the system keeps the conversation's context across a follow-up, and the clarification class requires the assistant to ask which metric is meant rather than guess. Of these, 100 refusal scenarios ask for something the assistant must decline, such as customer revenue, engineer email addresses, or a forecast stated with certainty. These are reported separately (Section 3.4).

| Need class | $n$ | Asked, and decided by |
|---|---|---|
| Factual table question | 350 | One number from the warehouse, such as a count of CI runs or pull request reviews. Decided by a frozen SQL query. |
| Multi-turn analysis | 150 | A question, then a follow-up that changes the repository, week, or filter. Decided by frozen queries for both turns. |
| Interface state | 150 | A question asked through the dashboard with the interface transition recorded. Decided by the value plus an error-free interface (139), or a blank input that must not error (11). |
| Formatting and time | 100 | A weekly series as a chart or table (75), or an answer within a stated time (25). Decided by the values plus the shape or the timing. |
| Document plus warehouse | 26 | A target in an attached document reconciled with the warehouse value, both labelled. Decided by a fixture and a query. |
| Clarification | 20 | An ambiguous request, such as "the health number" for a repository. Decided by the policy rule: ask which metric is meant. |
| Document question | 18 | Owner and cadence records read from an attached document. Decided by the fixture. |

Table 1: The 814 scenarios in the development set's primary scoring set, by need class.

### 3.2 Scenarios and evidence

Each scenario records the question or interface action, the expected outcome, the persona asking, a

risk class, and the evidence source that decides it. The evidence source is one of four kinds: a frozen DuckDB query over a frozen copy of the warehouse that returns the expected record, a fixture holding the fields extracted from a test document, a policy rule stating that a request must be declined (or when the metric is ambiguous, answered with a clarifying question), or an interface contract stating that the recorded dashboard transition must complete without error. A SHA-256 hash of the scenario content identifies each version, and comparing hashes between two runs shows whether a scenario changed in between (Section 5.3).

### 3.3 Deterministic checks

Deterministic checks decide scenarios where possible and constrain every review pass. The review judge decides the rest, which covers most test-set scenarios (Section 5.1). We call the stored check for a scenario its oracle. A factuality oracle compares the answer with the number or record returned by the frozen query. A policy oracle checks that a forbidden request was declined or an ambiguous one met with a clarifying question. A state oracle checks that the interface completed the recorded transition without error. Presentation oracles check rendering, such as whether a chart appeared, and never count as factual passes: a bare chart, a table, a matching keyword, or the absence of an error is not by itself a correct answer. Given the same stored answer, evidence, and rule version, a check returns the same verdict every time.

### 3.4 Counting rules

Every score in this paper is a pass count over a fixed denominator, so a scenario that fails to execute counts as a miss rather than shrinking the denominator. On the test set every system is compared on identical items, and Section 5.3 reports the development scenarios whose content differs between runs. The 100 refusal scenarios are scored separately because a system that emits nothing passes all of them. In the development runs every system refused all of them apart from one infrastructure failure, so they are not tabulated. Clarification scenarios are answered by the application server with a fixed template without a model call, so every system passes them. We report model-eligible counts beside each total. Appendix A gives the full accounting.

### 3.5 Recorded review

Some answers state the expected value in prose which cannot be parsed by the deterministic check (for example, a percentage with the wrong number of decimal places). The platform then marks the scenario `review_required` and records the judge, scoring-method version, and verdict. The review instrument used for every score in this paper combines deterministic evidence with a headline check by `gpt-5.4-mini` at temperature zero, which reads the question, the expected value, and the full answer, accepts equivalent representations, and returns strict JSON. Every turn of a multi-turn scenario must support the expected value. A value stated with a caveat about its filter or population still counts as stated, because the rule tests the value and the absence of a competing value for the same metric and population. The judge's approval alone is not enough. The saved answer must also pass a deterministic check, state the expected value in prose, or contain the expected rows in a rendered table. The prose check excludes SQL, table content, and chart code. Without this evidence, the case remains unresolved and counts as a miss. Section 4.5 describes the human audit of these records.

### 3.6 Freezing and provenance

Each run records everything it needs to recompute its own score later: the product commit, the scenario manifest and oracle versions, the prompt hash, the data source, the provider-reported model identity, and the review instrument, with raw rows, checkpoints, and review records kept alongside (Appendix A). Freezing applies to a comparison, not to the platform's lifetime: a replacement model can run against the same scenarios, data, and tools with its own recorded configuration, as the seven baselines show, and if the scenarios or scoring rules change, the manifest or instrument version changes with them. LLM review is the one non-deterministic step: recording its verdict makes aggregation reproducible, but calling the judge again can change the verdict, and Section 5.1 reports how often it did.

## 4 Experiments

The case study makes two measurements: how seven models compare when given the same data tools and a plain prompt, and how much Interval's task harness adds on top of the bare model it runs.

| Step | Scenarios |
|---|---|
| *Development set* | |
| Frozen development comparison scope | 968 |
| less 100 refusal scenarios, scored separately | 868 |
| less 54 document scenarios set aside at the freeze | 814 |
| *Test set* | |
| Generated after the harness was frozen | 100 |
| less 3 keys of one template found ambiguous in review | **97** |
| of which model-eligible (not clarification) | 87 |

Table 2: Scenario accounting. The development set's primary scoring set is the 814; the test set results use the 97. Appendix A gives the detail.

Table 2 gives the scenario accounting. Every baseline runs under two prompt conditions, schema-only and schema-plus-registry (Section 4.1). We keep one run per system and condition, so we report fixed-denominator pass counts and exact paired tests rather than confidence intervals from repeated runs; the judge's rerun variability is measured on the test set (Section 5.1). Execution limits are in Appendix A.

### 4.1 Systems under test

**DS4+harness.** Interval's deployed assistant runs DS4 (`deepseek-v4-flash-0731`, native thinking on) inside the task harness: a tool loop with a custom system prompt, rule packs, skills, sub-agents, and four governed tools over a read-only DuckDB warehouse: SQL query, computation, analysis card, and data lineage. The rule packs include a metric registry that defines each business metric: what it counts, over what grain and scope, what to fall back to, and the exact aggregate SQL. The data layer enforces table access, person-level data policy, result limits, and a ban on forecasting. The harness was iteratively improved on the development scenarios.

**The baselines.** A baseline is a model given the same four tools over the same warehouse, the same execution limits on the test set (six baselines' development runs used a longer timeout), no retries, and a plain prompt, with no skills, personas, sub-agents, or recovery logic. Seven models were run: GPT-5.6 Sol and GPT-5.6 Terra at medium reasoning effort (hereafter Sol and Terra), Claude Fable 5.1, Claude Sonnet 5, and Claude Opus 5 with adaptive thinking at medium effort (Fable, Sonnet, and Opus), GLM-5.3 with native reasoning (GLM), and DS4 itself with native reasoning. GLM-5.3 is open-weight; it was served by Ollama Cloud and priced at its publisher's list price, not self-hosted. Tables and figures use the full names.

**Two prompt conditions.** Under the *schema-only* prompt a baseline sees the warehouse tables and columns; under the *schema-plus-registry* prompt it also sees the harness's metric registry. Comparing the two prompts for one model measures what the registry adds; comparing bare DS4 with DS4+harness measures what the harness adds under each prompt. On the test set the schema-plus-registry runs used the harness's own registry for every baseline, replacing an earlier run with an older copy.

### 4.2 Procedure on the development set

Each baseline ran the 968 development scenarios once per prompt against the approved manifest. The DS4+harness figure comes from a combined review record (the production record) that merges a development record and a confirmation record, two batches at one frozen commit (Appendix A). For this paper every system's development review cases were re-reviewed offline with the test-set judge and evidence check of Section 3.5. The deterministic outcomes stand, because the checker cannot be rerun on a saved answer, and answers whose record is too short to review count separately as undeterminable. Every system is reported on the fixed 814-scenario primary set (Section 5.3).

### 4.3 The test set

The test set evaluates new scenario content using the development template families. After freezing the harness, we generated 100 scenarios with repository names, weeks, and metric values absent from the development set: 50 factual table questions, 25 multi-turn scenarios, 15 formatting and time scenarios, and 10 clarification scenarios. We excluded document and interface tasks, which require additional fixtures and recorded screen transitions. Expected values came from the same frozen warehouse, and the manifest was frozen before execution.

One author reviewed all 29 templates and about 38 keys directly. Three activity-contributor scenarios had ambiguous keys because the requested population allowed two interpretations. Removing

them left 97 scenarios. Every system had passed those cases except Terra, which missed one under the registry prompt. Ambiguous keys were the only reason for exclusion.

We also adopted two scoring rules. First, clarifying questions pass on three review-count scenarios under the registry prompt, because that prompt explicitly instructs models to clarify the meaning of "review." The schema-only prompt gives no such instruction, so those answers must match the review-event count. Scores excluding these scenarios appear in Appendix Table 6. Second, a checker's contradiction flag is overridden when it concerns a rounded or contextual number and both the judge and evidence check accept the answer. This affected three baseline cases and applied equally to every system.

The exclusions and both scoring rules were decided after inspecting results, by an author who knew the system identities. Each system ran separately, and all `review_required` cases used the method in Section 3.5. The headline results use the 97 retained scenarios.

### 4.4 Cost accounting

We estimate cost from the recorded model usage on the test set's 87 model-eligible scenarios. Cost per correct answer is the total cost divided by the number of correct answers on those same scenarios. These estimates are not reconciled against invoices.

For the five proprietary models, we use the runner's cost calculation for each call at the provider's published rates, including cache discounts. For GLM, we apply z.ai's list rates to the recorded tokens: $1.40 per million uncached input tokens, $0.26 per million cached input tokens, and $4.40 per million output tokens (Z.ai, 2026). Ollama Cloud served the GLM runs, so using z.ai's cache discount is a pricing assumption, not a measurement of the Ollama bill.

For DS4, we price the recorded tokens at Ollama Cloud's list rates for the tested checkpoint: $0.22 per million input tokens and $0.66 per million output tokens, without a cache discount (models.dev, 2026). Ollama Cloud served the production record. Table 4 also applies DeepSeek's lower off-peak tariff to the same bare DS4 tokens and projects self-hosted costs from measured throughput. Discounting GLM's cached input while applying no cache discount to DS4 reduces GLM's estimated cost relative to DS4. Appendix A gives the tariff sources and token accounting.

The headline figure is DS4's self-hosted model-serving cost, projected from throughput measured on our hardware, a server of about $36,000 (two RTX Pro 6000 GPUs serving vLLM (Kwon et al., 2023)), assuming continuous utilisation: we reran DS4+harness on the 814 development scenarios against one such replica and measured its busy time and throughput. The hourly cost is the server amortised straight-line over five years, $0.82, plus a 1.35 kW draw at the 2025 US average industrial electricity price of 8.62 cents per kWh (U.S. Energy Information Administration, 2026): $0.94 per hour, or $1.11 at California's 2024 industrial average of 21.53 cents (U.S. Energy Information Administration, 2025). For each test-set row we convert recorded tokens to busy time at the measured throughput, as a range between full and no overlap of prefill and decode (Appendix A).

### 4.5 Human audit

One author audited a stratified random sample of development-set answers: 60 of the 725 DS4+harness passes that went through review, and five failures from each of the seven baselines under the schema-plus-registry prompt, to check whether the checker marks correct baseline answers wrong. All 60 sampled DS4+harness passes were acceptable. Of the 32 baseline failures the auditor could decide, 14 were correct answers, and three more were marked unsure (Appendix A). These are observed counts only: the sample covers neither every stratum nor baseline passes, so it supports no population bound or corrected score. The audit was not blind to the system.

A second audit covered the test set, blind to the system: 40 saved registry-prompt answers in random order, with system and verdict removed, graded by one author against the key before the rules of Section 4.3 were decided: every registry-arm miss on the 97 (20) and 20 judged passes drawn at random. All 20 sampled passes were correct. The author initially judged 4 of the 20 misses correct. One now passes under the revised scoring rules. Two remain misses under the task requirements, and one audit judgment was mistaken. The author also rejected all ten clarification answers, which the final scoring accepts because the registry instructed models to clarify. Appendix Table 6 reports scores without those scenarios. Individual cases are detailed in Appendix A.

| | Schema-only | | With registry | |
|---|---|---|---|---|
| System | /97 | /87 | /97 | /87 |
| DS4+harness | | | 96 | 86 |
| DS4 (bare model) | 89 | 79 | 94 | 84 |
| GLM-5.3 | 95 | 85 | 96 | 86 |
| Claude Sonnet 5 | 94 | 84 | 96 | 86 |
| Claude Opus 5 | 97 | 87 | 97 | 87 |
| GPT-5.6 Terra | 92 | 82 | 94 | 84 |
| GPT-5.6 Sol | 96 | 86 | 97 | 87 |
| Claude Fable 5.1 | 95 | 85 | 97 | 87 |

Table 3: Test set: passes on the 97 scenarios kept after the template review (Section 4.3), and on the 87 of them that are model-eligible (the 10 clarification scenarios are answered by a server template, and every system passes them). One retained run per system and prompt, one final scoring instrument for every row. DS4+harness uses its own prompt, so it has one score.

# 5 Results

## 5.1 The test set

On the 97 test scenarios (Table 3, Figure 2a) DS4+harness passes 96. With the registry in its prompt, bare DS4 passes 94 and the six frontier models pass 94 to 97. Without the three review-count scenarios (Section 4.3), the frontier models pass 91 to 94 of 94, bare DS4 passes 91, and DS4+harness passes 93. Without the registry, the frontier models pass 92 to 97 and bare DS4 passes 89. The harness lifts bare DS4 from 89 to 96: DS4+harness passes 8 scenarios that bare DS4 fails, and bare DS4 passes 1 that it fails (exact McNemar test, $p = 0.039$; by template family, 4 vs 1, $p = 0.38$). With the registry in its prompt, bare DS4 and DS4+harness differ on 3 against 1 scenarios ($p = 0.625$): the registry alone recovers 5 of the 7 points. The frontier models do not need the registry on this test.

The misses are few enough to read one by one. Under the registry prompt, ten answers asked for clarification, as the registry instructed, and therefore counted as passes (Section 4.3). The frontier models five remaining misses involved conflicting metric and data-access rules, incomplete responses, execution failures, or different interpretations of a time constraint. None involved an incorrect value stated alone. The appendix details selected cases. The one DS4+harness miss used the wrong repository scope. Bare DS4's three misses involved repository scope or interpretation of a time constraint. A context-to-adapter method such as the Internalizer (Devine et al., 2026) could be used to help models recognize company-specific names.

The review judge decides 81 to 87 of each system's 97 scenarios. Repeating the review at temperature zero changed 1 verdict across all runs. The reported scores use the first review.

## 5.2 Cost

Under the schema-plus-registry prompt (Table 4, Figure 2b), the frontier models cost \$0.0111 (Sonnet) to \$0.0766 (Fable) per correct answer. Bare DS4 costs \$0.0043 at the Ollama Cloud list price and an estimated \$0.0010 to \$0.0018 self-hosted, so the frontier models cost an estimated 6 to 77 times as much as self-hosted DS4 under the common registry prompt, and about 3 to 54 times under the generally cheaper schema-only prompt. Against the list-price figure the registry-prompt multiple is 3 to 18, and DS4+harness costs about the same as bare DS4. The controlled comparison holds the prompt fixed; a deployer would run each frontier model in its cheaper condition, which narrows but does not close the gap. At California's electricity price the multiples fall by about 16%. The self-hosted figures are projections: accuracy comes from the gateway runs, throughput from a development-set rerun on one replica (Appendix A), and the cost covers model serving only, excluding the application host and engineering time.

DeepSeek's own off-peak tariff, applied to the same recorded tokens under the registry prompt, is estimated to give \$0.0030 per correct answer, or 1.6 to 3.0 times the self-hosted range.

The self-hosted range rests on the DS4+harness rerun of the 814 development scenarios on one two-GPU replica of Interval's own cluster, with ten concurrent workers and every prompt recomputed: 0.70 busy replica-hours, \$0.66 in total, and \$0.00082 per correct answer, against \$2.89 for the same tokens at the list price, 4.4 times more. Self-hosting assumes the replica is kept busy: it breaks even with the list price at about 21% utilisation if an idle server draws no power, and at about 23% if it draws full power while idle (Appendix A).[1] Every API figure is computed from recorded tokens, not from invoices (Section 4.4).

| System | Cost basis | Schema + registry Total | Per correct | × DS4 self-hosted | Schema-only Total | Per correct | × DS4 self-hosted |
|---|---|---|---|---|---|---|---|
| Claude Fable 5.1 | API usage, priced | $6.67 | $0.0766 | 42–77 | $4.58 | $0.0539 | 29–54 |
| Claude Opus 5 | API usage, priced | $3.71 | $0.0427 | 23–43 | $3.82 | $0.0439 | 24–44 |
| GPT-5.6 Sol | API usage, priced | $3.11 | $0.0357 | 19–36 | $1.33 | $0.0155 | 8–16 |
| GPT-5.6 Terra | API usage, priced | $1.59 | $0.0189 | 10–19 | $0.68 | $0.0083 | 5–8 |
| Claude Sonnet 5 | API usage, priced | $0.96 | $0.0111 | 6–11 | $0.61 | $0.0073 | 4–7 |
| GLM-5.3 | z.ai list | $1.19 | $0.0139 | 8–14 | $0.49 | $0.0057 | 3–6 |
| DS4 (bare model) | Ollama Cloud list | $0.36 | $0.0043 | 2–4 | $0.16 | $0.0021 | – |
| DS4 (bare model) | DS API, off-peak | $0.25 | $0.0030 | 1.6–3.0 | $0.12 | $0.0015 | 0.8–1.5 |
| DS4 (bare model) | self-hosted | $0.08–0.15 | $0.0010–0.0018 | 1 | $0.07–0.10 | $0.0009–0.0013 | – |
| DS4+harness | Ollama Cloud list | $0.32 | $0.0038 | 2–4 | | | |
| **DS4+harness** | **self-hosted** | **$0.07–0.14** | **$0.0009–0.0017** | **1** | | | |

Table 4: Estimated cost on the test set: totals over the 87 model-eligible scenarios of the 97, and cost per correct answer among them (correct counts: Table 3, /87 columns). Self-hosted rows convert recorded tokens to busy time at the throughput measured on our hardware and the hourly cost of Section 4.4, as a range, assuming continuous utilisation. In both halves, the multiple divides the row's cost per correct by the self-hosted bare DS4 range under the registry prompt, the condition in which bare DS4 reaches frontier-level accuracy; that row is the denominator and is not bold. The bold row, DS4+harness self-hosted, is the recommended configuration; it costs about the same as bare DS4, and it has no schema-only arm. The DeepSeek off-peak row applies that tariff to the same recorded bare DS4 tokens; it does not represent another model run.

| System | Schema-only | Schema + registry |
|---|---|---|
| DS4+harness | | 808 |
| DS4 (bare model) | 620 (99) | 650 (102) |
| GLM-5.3 | 595 (99) | 645 (89) |
| Claude Sonnet 5 | 640 (96) | 648 (98) |
| Claude Opus 5 | 595 (98) | 627 (98) |
| GPT-5.6 Terra | 578 (78) | 631 (91) |
| GPT-5.6 Sol | 623 (90) | 629 (92) |
| Claude Fable 5.1 | 613 (97) | 646 (96) |

Table 5: Development set re-reviewed with the test-set judge and evidence check: passes out of 814 and, in brackets, answers undeterminable from their saved record, counted as non-passes. The DS4+harness figure includes its retry rescues; DS4+harness uses its own prompt, so it has one score. The 814 include 44 scenarios that read an attached document; DS4+harness passes 42 of them and the baselines 1 to 18, so those rows are not comparable across systems. No gap is drawn from these figures (Section 5.3).

### 5.3 The development set

The development set is where the harness was built, so it cannot validate it. Re-reviewed with the test-set judge and evidence check, DS4+harness passes 808 of the 814 (799 without its 9 retry rescues), and the baselines 627 to 650 under the schema-plus-registry prompt (Table 5). We draw no gap from these figures: the harness was tuned on these scenarios, many baseline answers are undeterminable from their saved records, and the audit of Section 4.5 found correct answers among recorded baseline failures, so the baselines' scores understate them by an unknown amount.

Scenario hashes also show how a self-built benchmark drifts toward the system under test. Of the 814 scenarios, 37 differ in content between the production record and the baselines' manifest, including 16 of the 20 clarification scenarios. The team that maintains the assistant also maintains the scenarios, and when a scenario fails the fix can go into the assistant or into the scenario's wording or expected value. Each edit is defensible on its own; together they move the test toward what the assistant already does. The content hash makes that drift measurable.

## 6 Discussion

**The platform.** LEDGER turns a company's requirements into versioned test scenarios and ties every score to its evidence. The same scenarios can test a model replacement, a prompt revision, or a change to the harness, and recorded review verdicts make the judge's rerun variability measurable. These controls make a comparison inspectable; they do not remove development exposure or review uncertainty, which is why the test set exists.

[1]At 21% utilisation the replica is busy about 5 hours a day; the rest can serve batch jobs such as evaluation runs at no extra hardware cost, a value not counted here.

**Where the accuracy comes from.** Given the company's metric definitions, seven models from four providers land in a narrow range on this test, so on these tasks the choice of model moved accuracy little. The definitions matter most for the open-weight model: without the registry, bare DS4 trails the frontier models, and the registry alone carries most of the harness's lift, while the frontier models score about the same with or without it. What separates the candidates is cost: an estimated 6 to 77 times per correct answer. The comparison is between deployed system designs, not a ranking of base models, and it does not say how another open model would fare.

**The human effort behind the answer key.** No person wrote an expected answer: every expected value is a frozen query result, a hand-entered fact, or a policy contract, and a script fills the question templates (Section 3.1). The human work sits around that script: curating the agent-drafted template families, entering the fixture facts, making about a thousand admission decisions, reviewing the test-set keys template by template, and auditing samples of verdicts and answers. We did not log the hours. The work involved reviewing templates and admitting scenarios rather than writing a thousand answers by hand. A customer that replicates it needs a frozen copy of its data, a registry of what its metrics mean, templates over both, and one person with authority over what counts as correct.

## 7 Limitations

1. **Near the ceiling, single runs.** Every system is near the ceiling on the test set and each configuration has one retained run, so the test cannot rank the systems or establish equivalence.

2. **Template dependence.** Scenarios from one template share its query logic, so the scenario-level paired tests overstate the evidence; Section 5.1 gives the template-family counts.

3. **Review-count scenarios.** Under the registry prompt a clarifying question passes the three review-count scenarios, because the registry instructed it, so those scenarios don't separate systems in that arm; Appendix Table 6 gives the scores without them.

4. **Audits by an author.** The key review and answer audits were by one author; the test-set audit was blind to the system but not to the study. They check consistency and are not an independent error estimate.

5. **Platform scope.** Reuse is shown on one company's tasks and on tool-calling models. Task-specific oracles must still be written and validated, and confidential source data limit external reproduction.

## 8 Future work

- **A harder test set and an ablation.** A test set below the frontier models' ceiling, extended to the interface classes with human-reviewed admission, repeated runs, and metered power on the self-hosted replica, so that the comparison can rank models; then remove the harness's components one at a time (prompt, rule packs, skills, sub-agents) to measure what each contributes.

- **Adapters for accuracy and speed.** Hypernetworks that map context to a LoRA adapter (Devine et al., 2026) could improve accuracy by encoding company-specific definitions and names. This could help address repository-scope errors (Section 5.1). Encoding that context in adapters could also reduce prompt length and improve speed. Both effects could be measured with the same test. Question answering over attached documents is the subject of a companion paper in preparation.

- **From measuring the system to improving it.** Make the improvement loop routine across tasks and business domains, with independently checked oracles and the same human gate on what counts as correct.

- **Other open models.** Run a second open model through the task harness to test whether the harness's effect transfers.

## 9 Conclusion

LEDGER gives a company a test of its own AI assistant that survives changes to the assistant: fixed scenarios, deterministic checks, recorded review, and scores that trace to evidence. For Interval, that is what made building on a privately hosted open-weight model defensible: because every failure traced to a scenario, an expected value, and a

rule, the team could improve the system around the model on the development set, then show on 97 held-out scenarios that it reaches frontier-level accuracy at a projected fraction of the cost per correct answer. Without such a test, the safe choice is to stay on a frontier model and pay its price. With it, the choice of model becomes an engineering decision the company can measure, revisit, and defend.

## Author contributions and acknowledgements

B.S. designed and built the evaluation platform, ran the evaluations, reviewed the test-set question templates and answer keys, audited the development-set answers, and wrote the paper. P.A.D. contributed to the platform's design, consulted on research direction, and reviewed drafts of this paper. N.R. consulted on research direction and evaluation design, and reviewed drafts of this paper. A.C. co-created the concept of LEDGER and the case study with B.S., supervised the work, and exported the enterprise data corpus from Interval's development process.

## References

Peter Belcak, Greg Heinrich, Shizhe Diao, Yonggan Fu, Xin Dong, Saurav Muralidharan, Yingyan Celine Lin, and Pavlo Molchanov. 2025. Small language models are the future of agentic AI. *arXiv preprint arXiv:2506.02153*.

DeepSeek. 2026a. Models & pricing. https://web.archive.org/web/20260925215628/https://api-docs.deepseek.com/quick_start/pricing/. Internet Archive capture of 25 September 2026; accessed 16 October 2026.

DeepSeek. 2026b. Models & pricing. https://api-docs.deepseek.com/quick_start/pricing/. Live page, accessed 16 October 2026.

DeepSeek-AI. 2026. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*.

Peter A. Devine, Nick Ryan, Ben Sirb, and Alex Chiocchi. 2026. Internalizer: Portable context-to-parameter mapping for very large language models. *arXiv preprint arXiv:2610.11715*.

Liming Dong, Qinghua Lu, and Liming Zhu. 2024. AgentOps: Enabling observability of LLM agents. *arXiv preprint arXiv:2411.05285*.

Akshat Gupta, Jermaine Lei, Alexander Lu, Gopala Anumanchipalli, and Leshem Choshen. 2026. Automated discovery has no universally superior harness. *arXiv preprint arXiv:2607.18235*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pages 611–626.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, et al. 2025. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *International Conference on Learning Representations*. arXiv:2411.07763.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, et al. 2023. Can LLM already serve as a database interface? a big bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36.

Evan Miller. 2024. Adding error bars to evals: A statistical approach to language model evaluations. *arXiv preprint arXiv:2411.00640*.

models.dev. 2026. DeepSeek V4 Flash 0731. https://models.dev/models/deepseek/deepseek-v4-flash-0731/. Accessed 7 October 2026.

OpenTelemetry Authors. 2026. Semantic conventions for generative AI systems. status: Development, accessed 2026-09-09.

Oscar Sainz, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. 2024. Judging the judges: Evaluating alignment and vulnerabilities in LLMs-as-judges. *arXiv preprint arXiv:2406.12624*.

UK AI Security Institute. 2024. Inspect: An open-source framework for large language model evaluations. https://inspect.aisi.org.uk/. Accessed 16 October 2026.

U.S. Energy Information Administration. 2025. Electric power annual 2024, Table 2.10: Average price of electricity to ultimate customers by end-use sector, by state, 2024 and 2023. https://www.eia.gov/electricity/annual/html/epa_02_10.html.

U.S. Energy Information Administration. 2026. Electric power monthly, Table 5.3: Average price of electricity to ultimate customers.

https://www.eia.gov/electricity/monthly/. February 2026 release; preliminary 2025 annual U.S. industrial average, also reported at https://www.eia.gov/energyexplained/electricity/prices-and-factors-affecting-prices.php.

Apoorv Vyas, Aditya Dhanuka, Sina Khoshfetrat Pakazad, and Henrik Ohlsson. 2026. Data intelligence agents: Interpreting, modeling, and querying enterprise data via autonomous coding agents. *arXiv preprint arXiv:2606.19319*.

Yuxuan Wang, Jie Zhang, Tianyu Cai, et al. 2026. From agent traces to trust: A survey of evidence tracing and execution provenance in LLM agents. *arXiv preprint arXiv:2606.04990*.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2024. $\tau$-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Jiayi Yuan, Hao Li, Xinheng Ding, Wenya Xie, Yu-Jhe Li, Wenqian Zhao, Kaiyu Wan, Jiawei Shi, Xia Hu, and Zirui Liu. 2025. Understanding and mitigating numerical sources of nondeterminism in LLM inference. *arXiv preprint arXiv:2506.09501*.

Z.ai. 2026. Pricing. https://docs.z.ai/guides/overview/pricing. Accessed 13 October 2026.

Jun Zhang, Kevin Surrao, Teerapat Nitayanont, et al. 2026. DI-Bench: Systematically generating in-domain data intelligence benchmarks for enterprise agents. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing: Industry Track*. arXiv:2609.05776.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36.

## A Reproducibility

The evaluation uses confidential business data. Frozen manifests, included and excluded identifier lists, prompts, scoring rules, raw outputs, and audit records are retained for verification, resource locations and private scenario content are omitted, and external reproduction requires a shareable dataset.

**Scenario accounting.** The source pool holds 1,000 scenarios. The comparison scope was frozen at 968; the 32 outside it (30 policy and two document cases) were admitted to the pool separately and are not scored here. Of the 968, the 100 refusal scenarios are reported separately (Section 3.4). Every system refused all 100 under both prompts except Sol under the registry prompt at 99 of 100, the miss being an infrastructure failure. A further 54 document scenarios were set aside before any system was scored, because their answer keys had not completed human review when the comparison was frozen; they passed that review on 11 September 2026 and are not scored here. The remaining 814 form the primary scoring set. On the test set, dropping the three ambiguous keys lowers every score by 3, except Terra's under the registry prompt, which falls by 2 because Terra failed one of the three.

**Production record.** The production record, the development-set run of DS4+harness, was two batches over a disjoint partition of the 1,000 scenarios, 800 then a locked 200, both at the same frozen source commit with the same scoring scope and no source change between them. They contribute 629 and 185 of the 814. The 100 refusal and 32 strict-only scenarios all fall in the first batch. Locked means the source was frozen, not that the scenarios were held out. Nine scenarios passed on the runner's retry after a failed first attempt.

**Models and execution.** The DS4 checkpoint is `deepseek-v4-flash-0731`, served through Interval's gateway on the test set for both DS4+harness and the bare baseline. The frontier baselines are `gpt-5.6-sol` and `gpt-5.6-terra` (reasoning effort medium), `claude-fable-5-1`, `claude-sonnet-5`, and `claude-opus-5` (adaptive thinking, effort medium), and `glm-5.3` (native reasoning). Every test-set run used a 170-second timeout per scenario, a limit of 40 tool calls, zero model retries, ten parallel workers, and one system at a time. The baselines' development runs used a 600-second timeout for six of the seven models. The DS4+harness and baseline checker

versions differ only in legacy interface checks that no test scenario uses. Superseded runs are retained but excluded from the reported results. The judge (OpenAI `gpt-5.4-mini`) and the screenshot OCR (Ollama Cloud) are external services that receive evaluation data.

**Cost model.** The recorded DS4 tokens are repriced at the Ollama Cloud list price for the tested checkpoint, `deepseek-v4-flash:0731`, on the API that served the production record: $0.22 per million input tokens and $0.66 per million output tokens, with no cache-hit discount (models.dev, 2026). DeepSeek's own Flash-tier tariff is lower, $0.15 and $0.60 off-peak and double at peak; its current page names V4.1-Flash, and an archived copy applies the same price to the retired `deepseek-v4-flash` name (DeepSeek, 2026a,b). At that tariff the list-price rows fall by about 30%. GLM's runs through Ollama Cloud report cached prompt tokens, which the runner records as the difference between total and uncached tokens: under the registry prompt, 0.17M uncached and 1.26M cached input tokens over the 87 model-eligible scenarios (0.46M cached under the schema-only prompt). The cached tokens are priced at $0.26; at the full input price GLM would cost $0.031 per correct answer under the registry prompt, 17–31 times self-hosted DS4. The five proprietary rows are the runner's cost from each call's recorded usage at the provider's rates, which include the provider's cache discounts; DS4 is priced with none. The cached-token count is the runner's residual of total over uncached input and output tokens, which may include cache writes. Sonnet's registry-prompt row has 86 cost records: its one failure made no model call, so it adds no cost and no correct answer. Under the registry prompt, bare DS4 recorded 1.52M input and 0.043M output tokens over the 87 model-eligible scenarios, and DS4+harness 1.35M and 0.043M.

**Self-hosted measurement.** On 7 and 8 October 2026 we reran DS4+harness against one replica of DS4 on Interval's own cluster: two RTX Pro 6000 GPUs serving vLLM with tensor parallelism 2. The rerun used the production record's runner, commit, arguments, and ten workers on the 814 primary scenarios, with matching application, prompt, scenario, and manifest hashes. Every request carried a fresh cache salt, so every prompt was recomputed, as the API pricing assumes. The run made 1,944 model calls with no failures or retries, at 11,927,565 input and 402,306 output tokens and an aggregate 4,720 prefill and 159 decode tokens per second. Its serving settings (reasoning effort high, fp8 KV cache, speculative decoding) may differ from the record's. At most 11 requests ran on a replica that admits 20, and queueing counts as busy time; both make the figure conservative.

Cost is the time the replica had at least one of our requests in flight, times the hourly cost of Section 4.4 (NVIDIA's list price for the cards plus a platform that runs them at full PCIe 5.0 bandwidth). At California's industrial price the rerun costs $0.78, or $0.00098 per correct answer; the list price on the same tokens is then 3.7 times higher, and self-hosting breaks even at about 22% utilisation (27% if an idle server draws full power). For a test-set row, busy time lies between the larger of input tokens over the prefill rate and output tokens over the decode rate (prefill and decode overlapping, as measured) and their sum (no overlap). At the California price the bare DS4 range is $0.0012 to $0.0022 per correct answer.

**Adjudication.** The record's evidence audit credits a number found anywhere in the answer, which can credit contradictory, negated, and null answers, so we re-adjudicated every review case under a stricter policy: a number passes only if the answer states the expected value as its answer to the requested metric and states no other value for it. A value mentioned only as a rejected alternative is not support. When the question says to use zero when nothing matched and the key is zero, a null answer fails, and records must match one table with one consistent column mapping, as the leading block of the series. A deterministic pass flagged doubtful turns, an independent judge (Claude Sonnet 5) reviewed every flagged and failed turn, and a person reviewed every judge failure and disagreement. Under this policy the self-hosted rerun moves from 803 to 800 of 814. The DS4+harness development-set figure we report (Section 5.3) is the re-review with the test-set judge and evidence check, 808 of 814. The record's retry rescues pass under the stricter policy. Three test-set answers state the expected value with a caveat and were adjudicated separately from the blind sample: Opus's reviewed-pull-request counts for two repositories, called upper bounds because the bot filter could not run, and Sonnet's reviewer count for one repository, given

as one with "0 or 1" human reviewers; the warehouse confirms each value under either population, so all three pass under the review rule.

**Audit sampling.** The development-set audit sampled 60 reviewed DS4+harness passes from categories containing 723 of its 725 review passes. It also sampled five registry-prompt failures from each baseline. The sample did not cover all categories or include baseline passes, so the findings do not estimate an overall scoring-error rate.

**Test-set audit cases.** The author initially judged 4 of the 20 misses correct. The final decisions were:

1. **Opus: now a pass.** Its rounded unit conversion is accepted under rule 2.
2. **GLM: remains a miss.** It interpreted "within 150 seconds" as a filter on run duration rather than a response-time limit. Six of eight systems used the intended interpretation. All eight led with the expected total on the three related scenarios.
3. **Terra's table: remains a miss.** It returned the correct 26-row table but omitted the final sentence, so it counts as a completion failure.
4. **Terra's withheld count: remains a miss.** The saved answer contains only a blocked query and states no value. The author's initial audit judgment was mistaken.

The author judged all ten clarifying questions on the review-count scenarios unreasonable answers, since the question names review events; rule 1 passes them because the registry instructed the clarification, and Appendix Table 6 gives every score without those scenarios.

**Scoring rules adopted after inspection.** After inspecting every test-set miss, we adopted two scoring rules (Section 4.3). These decisions were made with system identities known.

First, clarifying questions pass on three review-count scenarios under the registry prompt, because that prompt explicitly instructs models to clarify. Second, a checker contradiction caused by a rounded or contextual number does not count as a failure when both the judge and evidence check accept the answer.

These rules increased four frontier models' registry-prompt scores and bare DS4's schema-only score. They did not change any DS4+harness score. The schema-only comparison between DS4+harness and bare DS4 changed from $p = 0.012$ to $p = 0.039$.

| | Schema-only | | With registry | |
|---|---|---|---|---|
| System | /94 | /84 | /94 | /84 |
| DS4+harness | | | 93 | 83 |
| DS4 (bare model) | 86 | 76 | 91 | 81 |
| GLM-5.3 | 92 | 82 | 93 | 83 |
| Claude Sonnet 5 | 91 | 81 | 93 | 83 |
| Claude Opus 5 | 94 | 84 | 94 | 84 |
| GPT-5.6 Terra | 89 | 79 | 91 | 81 |
| GPT-5.6 Sol | 93 | 83 | 94 | 84 |
| Claude Fable 5.1 | 92 | 82 | 94 | 84 |

Table 6: Post-hoc sensitivity, decided after seeing the results: the test set without the three review-count scenarios, on which a clarifying question passes under the registry prompt and fails under the schema-only prompt (Section 4.3). Passes on the remaining 94 scenarios and on their 84 model-eligible scenarios, from the saved outcomes of the retained runs. DS4+harness uses its own prompt, so it has one score. The primary results are in Table 3.

**Review-count sensitivity.** Table 6 gives the test-set scores without the three review-count scenarios, an analysis decided after seeing the results; the paired counts do not change.