

BV-SALA Savings Layer: Verifiable Measurement of Cost Savings in Large Language Model Applications

BiVELIO RESEARCH TEAM

BiVelio Inc — savings.bivelio.com

October 2026

Abstract—Inference cost has become a dominant expense line in LLM-based software engineering, yet optimization decisions are routinely justified by point estimates without confidence intervals or metrological traceability. This paper presents BV-SALA Savings Layer, a measurement and cost-optimization layer that values every saving exclusively with the provider’s own billing counter, splits gains into four ledgers that are never summed, and publishes every public-bench trial with its 95% confidence interval. On a stream with 30% exact repeats (GSM8K), the SDK reduced the `gpt-4o-mini` bill by 31.5% (95% CI: 25.7–36.7) and `claude-sonnet-5` by 75.1% (95% CI: 69.3–79.6), with the provider’s own cache discount reported separately. We formalize the double-run methodology, define measured-versus-estimated accounting, derive the per-seat break-even threshold, and discuss threats to validity in production.

Keywords—LLM; inference cost; metrology; prefix caching; confidence intervals; prompt optimization.

I. INTRODUCTION

The mass adoption of large language models (LLMs) has added a new and volatile item to engineering budgets: the inference bill. Unlike other infrastructure costs, LLM spending accrues per request and hinges on design decisions that are barely visible — the format of the data injected into the prompt, the re-reading of conversational context from one turn to the next, the provider’s caching policy, or the share of repeated requests — any of which can multiply the unit cost of the same feature by orders of magnitude. The literature has proposed several lines of attack: prompt compression [1, 2], attention reuse and prefix caching [3–6], semantic caches [7], routing across models [8], and efficient inference servers [9]. From the standpoint of the organization that pays, however, they all share a common deficit: the absence of a *verifiable measurement of savings*, that is, a procedure that confirms the gain with the very instrument that issues the bill.

In practice, the savings figures that circulate in commercial materials rarely include the three elements that make them interpretable: the confidence interval, the specific model on which they were measured, and the characterization of the test traffic. An isolated percentage — without dispersion, without a model, and without a workload — allows no extrapolation, and it can blend gains of a different nature into a single number: calls that were never executed, tokens that were removed, tokens that were compressed, and discounts that the provider was already applying through its own prefix cache. This mixture, besides inflating the result, obstructs decision-making: a lever that eliminates calls (whose effect depends on how repetitive the traffic is) is not equivalent to one that rewrites structured data (whose effect depends on the shape of the data).

This paper presents and formalizes the methodology of BV-SALA Savings Layer, a measurement and cost-optimization layer for LLM-based applications developed by BiVelio. The design rests on a principle of bookkeeping hon-

esty: *only what the provider’s counter confirms is counted*. The contributions are the following:

1. **Non-intrusive measurement architecture** (Section III): an in-process SDK for Node.js applications and a local *gateway* for coding agents that keep traffic direct between the application and the provider; prompts never transit through third-party servers.
2. **Separate-ledger accounting** (Section IV): savings are decomposed into four ledgers (avoided calls, eliminated tokens, compressed tokens, and provider-cached tokens) that are reported separately and never summed into a single marketing figure.
3. **Double-run protocol against the real provider** (Section IV): every scenario is measured twice — with and without the layer — and both runs are valued with the provider’s official counter, with 95% confidence intervals and an explicit classification between measured and estimated.
4. **Public bench and economic analysis** (Sections V and VI): individual publication of every trial — including negative or non-publishable results — and a break-even threshold model that relates the per-seat fee to the repeat rate of the organization’s own traffic.

The scope of the data analyzed here must be fixed from the outset. All savings figures come from the product’s public bench (published and verifiable in savings.bivelio.com), measured against real providers with their billing counters; the economic scenarios of Section VI are illustrative projections built from those figures and are identified as such. The authors participate in the development of the product described; the discussion section explicitly addresses the threats to validity that this origin implies.

II. RELATED WORK

Since models based on the Transformer architecture bill per processed token [10], inference-cost optimization has been pursued along four main fronts. The first is **compute reuse at the provider level**: Anthropic’s prefix cache bills repeated tokens at the beginning of the prompt at a substantial discount relative to the input price [5], and OpenAI applies an analogous mechanism of automatic discounting to requests with a common prefix [6]. On the academic side, Prompt Cache [3] reuses attention modules across requests and CachedAttention [4] persists the multi-turn KV-cache in an intermediate store for long conversations. These mechanisms reduce the bill without any client action, but their effect dissolves inside the final amount, and no client-side instrument isolates it.

The second front is **reducing the number of tokens**: LLMingua compresses prompts by removing low-information tokens [1] and *gist tokens* learn reusable context compressions [2]. The third is **response caching**: GPT-Cache introduces a semantic cache that serves previous answers to similar queries [7], with the attendant tension between savings and answer freshness. The fourth is **routing and model selection**: FrugalGPT orchestrates cascades of models of different cost to match the capacity each query demands [8], and inference servers such as vLLM [9] and speculative decoding [11] reduce the cost per served token.

BV-SALA Savings Layer does not compete with any of those levers: it wraps them. The layer sits between the application and the provider, activates a safe subset of optimizations (serving exact repeats without a new call, and rewriting the format of structured data) and, above all, *measures*: it gives each lever its own ledger and verifies every gain with the provider’s billing counter, including the discount the provider was already applying on its own, which is published separately from the savings attributable to the layer. Compared with the works cited above, whose evaluation focuses on latency or quality, the contribution here is metrological: published confidence intervals, measured/estimated separation, and traceability of savings down to the bill. This orientation is consistent with the guide to the expression of uncertainty in measurement [12]: a measurement result is not complete without its associated uncertainty.

III. SYSTEM ARCHITECTURE

BV-SALA Savings Layer materializes as three complementary components, whose deployment is illustrated in Figure 1. The cross-cutting criterion is that traffic between the application and the provider is never diverted: the components operate *in process* or *on the local machine*, and only token counts and amounts derived from the provider’s counter travel to BiVelio’s servers.

A. SDK for Node.js applications

The SDK (@bivelio/savings-layer) targets JavaScript and TypeScript applications that call an LLM API with their own key and pay per token. It installs as a single ESM package with *zero runtime dependencies* — the exact tokenizer is an optional peer dependency — so it introduces no

third-party code alongside the application’s own. Its use requires a single change: the call to the provider is wrapped with `createSavingsLayer`, which receives the provider and the license key, and every invocation to `generate()` or `generateStream()` returns, alongside the response, a `SavingsReport` object with the same content the dashboard reads. In *streaming* mode, the provider’s final counter is read when the stream closes; if a stream ends without that counter, the request is logged as *unmeasured* and never counts as measured savings, which prevents over-counting in production. The SDK is compatible with any endpoint compatible with the OpenAI API, adds native adapters for Anthropic and Gemini, and covers providers such as Mistral, DeepSeek, Azure OpenAI, Bedrock, Ollama, or vLLM.

B. Gateway for coding agents

The second component targets the other large mass of spending: coding agents (Claude Code [13], Codex CLI, Aider, Goose, among others) used over an API key. A single command (`npm i -g @bivelio/savings-layer` and `bvsala claude`) places a local meter in front of the terminal without changing any code or agent configuration. The gateway counts every request whose usage the provider reports, using the provider’s own counter and the models with a list price in its catalog, and accumulates spending in the dashboard per day and per model. With a PRO license on Anthropic models it applies by default a cache-policy lever: it moves cache writes from the one-hour TTL to the five-minute TTL only on turns where most of the prompt is new (the first turn of a session or a compaction), and leaves the chained turns of an ongoing session untouched. The lever changes the price that is billed, never the content the model reads, and its net effect — including the cache misses it causes — is what the bench publishes. For languages other than JavaScript, the official OpenAI or Anthropic SDK can point to the gateway as base URL, so usage logging remains under the same conditions.

C. Savings skill for subscriptions

For teams that consume LLMs through subscription plans with usage windows, there is a third, free piece: *savings-mode*, a *skill* of pure instructions (no proxy, no interceptor) that the agent installs in `~/.claude/skills` and that reduces the two masses that deplete the quota fastest: the model’s output and tool results. Its effect on subscription limits is pending a published A/B measurement and, consequently, is not associated here with any figure.

D. Privacy model

The deployment above sustains a simple guarantee: prompts do not leave the client’s infrastructure. The SDK wraps the call inside the application’s own process and the gateway measures on the local machine; the platform receives counts and amounts, not conversational content. This separation matters for adoption in regulated sectors and prevents the measurement layer itself from becoming a new data-leakage point, a risk that the semantic-cache literature has flagged for designs that centralize query content.

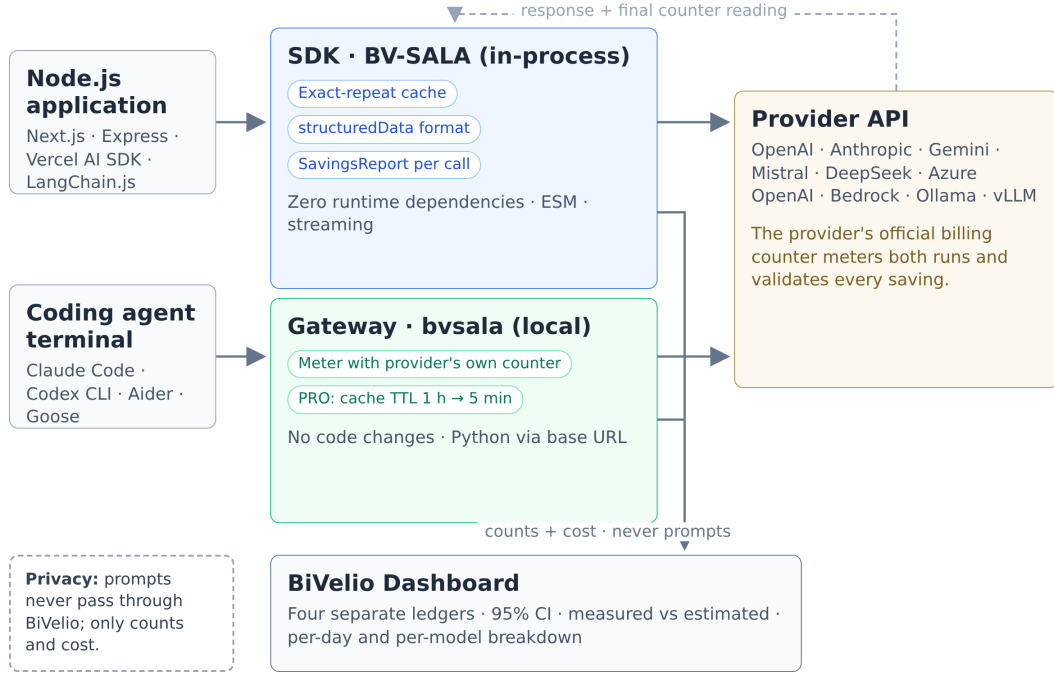


Figure 1. Deployment architecture of BV-SALA Savings Layer. The SDK wraps the call inside the Node.js application itself and the `bvsala` gateway places a local meter in front of the coding agent’s terminal; on both paths, traffic reaches the provider’s API directly and only counts and amounts (counts + cost) feed the dashboard, which separates the four ledgers and publishes the measurement together with its uncertainty.

IV. MEASUREMENT METHODOLOGY

A. Double-run principle

Every savings figure produced by the system is obtained with a double-run protocol against the real provider. For a load scenario $\mathcal{S}$ (for example, a GSM8K battery [14] with a fraction of repeated requests), two conditions are executed on the same model and the same content: the base condition $\mathcal{B}$, without the layer, and the instrumented condition $\mathcal{I}$, with the layer active. Let C_{base} and C_{opt} be the amounts that the *provider’s official billing counter* attributes to each condition. The primary metric is the relative reduction of the bill,

$$s = \frac{C_{\text{base}} - C_{\text{opt}}}{C_{\text{base}}}, \quad C_{\text{base}} > 0, \quad (1)$$

which the SDK exposes per request as `report.cost.netSavingsRatio`. The choice of instrument is not trivial: valuing savings with one’s own tokenizer or price card would introduce a systematic bias (tokenization models that differ from the one the provider bills, outdated prices, discounts not applied). By using the counter that issues the bill, the result is metrologically traceable to the amount the organization will actually pay, in the sense of traceability required by the uncertainty guide [12].

B. Measurement uncertainty

A point estimate s is not enough to decide. The public bench accompanies every result with a 95% confidence interval

built by non-parametric resampling (*bootstrap*) of the per-request costs measured in both conditions [15]: the estimator of equation (1) is recomputed over B resamples of the scenario’s requests, and the interval is read at the 2.5th and 97.5th percentiles of the resulting distribution. Denoting by $\hat{\sigma}_{\text{boot}}$ the standard deviation of those resamples, the normal approximation is

$$\text{CI}_{95\%}(\hat{s}) = \hat{s} \pm z_{0.975} \hat{\sigma}_{\text{boot}}, \quad z_{0.975} \approx 1.96. \quad (2)$$

The system’s editorial rule is explicit: *a percentage without its interval, without its model, and without its traffic means nothing*, so all three elements are published together for every bench trial.

C. The four ledgers

The `SavingsReport` splits savings into four ledgers that are reported separately and never summed (Table 1). Every avoided call is valued with the cost that the counter attributed to the identical request that was already executed, so the valuation of savings comes from the same instrument as the bill even though the call was not repeated. Amounts attributable to estimates (for example, valuing a cached token when the counter does not break it out) are flagged with their cost basis and reported in the *estimated* category, separate from the *measured* one; both categories are visible per request and aggregated.

The separation matters for two reasons. First, it avoids double counting: a token served by the provider’s cache can-

Table 1. The four ledgers of the SavingsReport. They are reported separately; summing them is deliberately avoided so that savings are not inflated by double-counting mechanics.

Ledger	Content and valuation criterion
avoided_calls	LLM, retrieval, or tool calls that were never executed; valued with the counter reading of the identical request already executed.
eliminated_tokens	Information removed from the prompt because it was not needed; valued with the provider’s counter.
compressed_tokens	Information kept but encoded with fewer tokens (e.g., rewriting of structured data).
provider_cached_tokens	Tokens that remain in the prompt but that the provider bills cheaper through its own prefix cache; this is the provider’s discount, not the layer’s, and it is reported separately.

not simultaneously be presented as the layer’s savings. Second, it guides the engineering decision: each ledger answers to a different lever (request deduplication, prompt hygiene, data format, and cache policy), with different dependencies on traffic and, therefore, different extrapolations.

D. Measured versus estimated

The system classifies every amount under one of two cost bases. An amount is *measured* when a reading of the provider’s counter supports it (end of stream in the SDK, reported usage in the gateway); it is *estimated* when it is valued with price cards or with the counter of an equivalent request. A stream without a final counter reading is logged as unmeasured and feeds no measured-savings figure. This discipline, together with the individual publication of every trial — including those that yield no publishable figure, such as the non-breakage verifications on Gemini — is what makes it possible to audit the system’s behavior against its own bill.

E. Public bench of trials

Measurement batteries run weekly against real providers and are published one by one in the public bench, with the full report of each trial: model, SDK version, traffic characterization, amounts of both conditions, resulting ledgers, and confidence interval. The bench serves a double purpose: external reproducibility (anyone can audit the protocol) and internal control (a lever whose effect has not been measured A/B cannot be announced). In the reference bench of September 28, 2026, the installed SDK’s savings came from the avoided-calls ledger and, on Claude, from the provider’s cache discount; the published package includes no compressor, so the compression ledger only activates with structured data.

V. PUBLIC BENCH RESULTS

Table 2 summarizes the trials with a published figure from the public bench, and Figure 2 plots them with their confidence intervals. In the OpenAI trial, a GSM8K stream [14]

Table 2. Results with a published figure from the public bench (measured with each provider’s counter; both members of each A/B pair are billed with the same instrument).

System and model	Savings	95% CI	Scenario
SDK · gpt-4o-mini (OpenAI)	31.5%	[25.7; 36.7]	GSM8K; 30% exact repeats; SDK 0.4.31
SDK · claude-sonnet-5 (Anthropic)	75.1%	[69.3; 79.6]	Same stream; 41 pp of provider cache
Gateway · Anthropic PRO (TTL)	2.5%	[1.5; 3.6]	51 Claude Code sessions; list prices

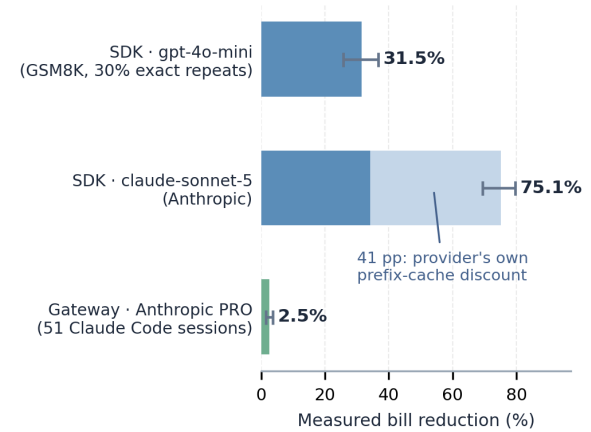


Figure 2. Bill reduction measured with each provider’s counter, with 95% confidence intervals. On *claude-sonnet-5*, the light portion corresponds to the provider’s own prefix-cache discount (41 percentage points of the original bill), which the system reports separately from the savings attributable to the layer.

in which 30% of the requests are exact repeats produced a bill reduction of 31.5% (95% CI: 25.7–36.7) on *gpt-4o-mini* with SDK version 0.4.31. The critical interpretive point is that the result is *not a forecast*: savings depend on how much repetition the organization’s own traffic contains. The empirical slope — approximately 1.05 bill points per repetition point, with interval [0.86; 1.22] — is what Section VI uses for the illustrative scenarios.

In the Anthropic trial on the same stream, *claude-sonnet-5* showed a reduction of 75.1% (95% CI: 69.3–79.6). The breakdown is essential to avoid overvaluing the result: about 41 percentage points of the original bill correspond to the discount that Anthropic’s own prefix cache was already applying [5], and the system reports it as the fourth ledger, not as attributable savings. The remainder comes from the avoided-calls ledger: repeated requests are served without a new call to the provider and are valued with the counter of the identical request already executed (Section C).

The third row corresponds to the gateway with a PRO license on Anthropic: the lever that moves cache writes from the one-hour TTL to the five-minute TTL on turns with a mostly new prompt produced a net saving of 2.5% (95% CI: 1.5–3.6), already net of the cache misses the lever itself causes on chained turns. The simulation was run at list prices

over 51 Claude Code sessions of a single organization (the team’s own), and the bench report itself delimits its scope: one organization, one provider, list prices, and a small saving that mainly serves to signal where Anthropic spending concentrates.

An illustrative case of editorial discipline is Gemini. The weekly batteries of October 1 and 2, 2026 ran the four complete batteries against Gemini 3.8 Flash and two partial runs against Gemini 3.1 Pro, with no failures of the layer and with no denial being cached or credited as savings; nevertheless, no savings figure is published for Gemini. The distinction between *non-breakage verification* (the layer breaks nothing and does not over-bill) and *savings measurement* (a published interval exists) is maintained even at the cost of being unable to show a number for a supported provider.

VI. ECONOMIC ANALYSIS AND SCENARIOS

A. Pricing model

The offering is structured in four plans (Table 3). The Measure plan is free and operates in *shadow mode*: the layer observes real traffic, changes no response, and the dashboard shows the savings it would have produced, valued with the provider’s counter. The paid plans are fixed per-seat fees — where a seat is a machine or service identity, not a person — and include no commission whatsoever on measured savings, which aligns the layer vendor’s incentive with the precision of the measurement rather than with its magnitude.

B. Break-even threshold

Let B be the monthly provider bill that traverses the layer, ρ the fraction of requests that are exact repeats, and P the per-seat fee; when a team contracts n seats at fee P (for example, $P = 15$ €/month on the Pro plan), adoption is profitable when the expected monthly savings exceed the fee:

$$nP < B \cdot s(\rho), \quad \rho^* = \frac{nP}{B \cdot s}, \quad (3)$$

where ρ^* is the minimum repeat rate for break-even, conditioned on the avoided-calls ledger and on the empirical slope s introduced in Section V. With $s = 1.05$ (point estimate) and its interval $[0.86; 1.22]$, a two-seat team ($nP = 30$ €/month) breaks even at $\rho^* \approx 5.7\%$ on a €500 bill, at $\rho^* \approx 1.4\%$ on €2,000, and at $\rho^* \approx 0.3\%$ on €10,000. Figure 3 plots these thresholds with the uncertainty band of the slope.

C. Illustrative scenarios

The following three readings are illustrative projections built with equation (3) and do not constitute measurements: (i) a product team with €500/month facing the provider and a 10% repeat rate expects avoided-calls savings of about €52/month (interval €43–61), an order of magnitude above the two-seat fee; (ii) a service with €2,000/month and only a 2% repeat rate expects about €42/month (€35–49), still profitable for two seats but already sensitive to the interval; (iii) a deployment with €10,000/month breaks even with less than a 0.3% repeat rate, so practically any traffic with natural deduplication (RAG over stable corpora, repeated evaluations, assistants with recurring templates) clears the fee with room

Table 3. Published plans and fees (VAT excluded; reverse charge for EU VAT-registered companies). A seat is a machine or service identity; CI runners and ephemeral identities do not count.

Plan	Fee	Scope
Measure	Free	Shadow mode; one organization, up to 5 seats and 20,000 requests/day.
Pro	€15/seat/month (or €150/year)	All levers, per-request receipt and dashboard; no commission.
Team	€129/month (or €1,290/year)	10 seats included; additional seat at €12/month.
Enterprise	Custom	Dedicated contract and negotiated billing.

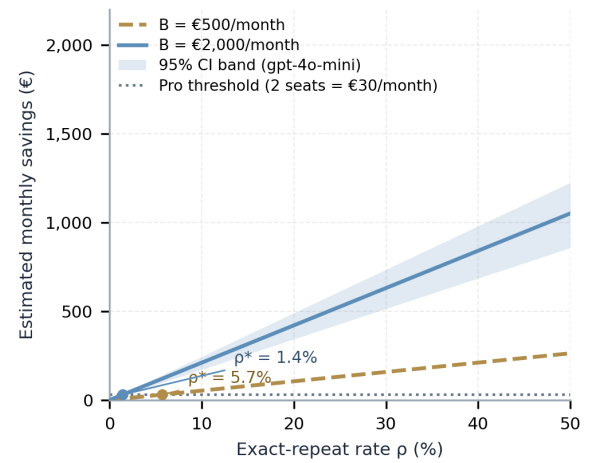


Figure 3. Illustrative scenario (not measured): expected monthly savings $S(\rho) = B \cdot s \cdot \rho$ from the avoided-calls ledger versus the exact-repeat rate ρ , for two monthly bills B . The band corresponds to the 95% CI of the gpt-4o-mini trial’s slope; the points mark break-even for a two-seat Pro plan (equation (3)).

to spare. Three caveats bound the generalization: the slope comes from a single model and a single stream; the gateway already showed that some levers have small effects (2.5% on Anthropic PRO); and the free shadow mode makes it possible to estimate ρ and the savings on one’s own traffic *before* any payment, which is the recommended path to replace this extrapolation with direct measurement.

VII. DISCUSSION AND THREATS TO VALIDITY

A. Internal validity

The main internal threat to any savings measurement is that the two conditions of the A/B pair do not compare the same thing: models upgraded mid-trial, prices changed, or different traffic across conditions. The double-run protocol mitigates this by executing both conditions against the real provider in contiguous time windows and with the same con-

tent, and by using the provider’s counter as the only valuation instrument in both branches. The trade-off is that the protocol does not isolate the provider’s own variations within the window; the *bootstrap* interval reflects dispersion across requests, not the temporal drift of prices. Shadow-mode use — where the base and instrumented conditions coincide physically and temporally, and only the counterfactual is computed — reduces this threat even further for the pre-purchase evaluation phase.

B. External validity

The published figures come from specific contexts: an academic stream (GSM8K) with a repeat rate fixed at 30% for the SDK trials, and — for the gateway — 51 sessions of a single organization at list prices. The savings of the avoided-calls ledger scale essentially with ρ , the repeat fraction of the organization’s own traffic, which varies enormously across applications: a customer-support chatbot with recurring templates may double or triple the benchmark stream’s ρ , while a creative-content generator may approach zero. No bench figure should be read as an individual forecast; the defensible extrapolation is the one that goes through measuring ρ in shadow mode on real traffic, which the free plan allows at no cost and with no change of behavior.

C. Privacy and dependence

The in-process, local-machine architecture removes the main objection to intermediate layers: prompt content never transits third parties, which eases adoption in environments with sensitive data. In exchange, the layer introduces a minimal but real operational dependence: the license is verified locally with a signature, and the package is a single ESM module with no runtime dependencies, which limits the supply-chain surface. Measure mode operates without touching any response, so the initial evaluation adds no functional risk; once the levers are activated, the *measured/estimated* logging and the exclusion of streams without a final counter prevent a faulty measurement from contaminating the accounting.

D. Limitations

Three limitations delimit the conclusions. First, the ledgers are not summed — by design — so no single figure summarizes the layer’s total effect; the reader must work with the breakdown. Second, the package includes no compressor: the compression ledger only activates with structured data that the application delivers as such, and its effect can be estimated in the browser tool before paying. Third, the gateway’s agent coverage is partial (Cursor is not measurable and Gemini CLI is not yet), and the subscription skill still lacks a published A/B measurement, so it is associated with no figure. Finally, the commercial origin of the work — a team evaluating its own product — demands treating the figures as declared and reproducible through the public bench, not as verified by a third party.

VIII. CONCLUSIONS AND FUTURE WORK

This paper has formalized the measurement methodology of BV-SALA Savings Layer, a cost-optimization layer for LLM applications whose guiding principle is bookkeeping

before algorithms: only what the provider’s billing counter confirms counts as savings. The architecture — a zero-runtime-dependency in-process SDK for Node.js and a local gateway for coding agents — keeps traffic direct between application and provider, exposing only counts and costs. The four-ledger accounting (avoided calls, eliminated tokens, compressed tokens, and the provider’s own cache discount) prevents inflating savings through double counting, and the double-run protocol with 95% confidence intervals (31.5% on gpt-4o-mini and 75.1% on claude-sonnet-5 over a stream with 30% exact repeats, and a net 2.5% from the gateway’s PRO lever) turns savings into a measurable, auditable, and comparable quantity. The break-even threshold model relates the per-seat fee to the repeat rate of the organization’s own traffic and delimits when adoption is justified; the free shadow mode makes it possible to replace that extrapolation with direct measurement before committing to a paid plan.

Future work is organized along three lines. The first is to widen the empirical base: trials with published figures on more providers (the non-breakage verification on Gemini is already underway, pending a published interval), more models, and third-party production traffic, so that the slope $s(\rho)$ stops depending on a single stream. The second is to complete the measurement cycle of the remaining pieces: the effect of the subscription skill through a published A/B trial, and the coverage of additional agents in the gateway. The third is methodological: characterizing the interaction between ledgers when several levers are activated together (for example, deduplication and cache policy simultaneously) and publishing the resampling battery that generates the intervals, so that third parties can reproduce the uncertainty without access to the original data.

Conclusiones (resumen en español)

Este trabajo ha formalizado la metodología de medición de BV-SALA Savings Layer, una capa de optimización de costes para aplicaciones basadas en LLM cuyo principio rector es contable antes que algorítmico: solo cuenta como ahorro lo que el contador de facturación del proveedor confirma. La arquitectura — SDK en proceso con cero dependencias en ejecución para Node.js y gateway local para agentes de código — mantiene el tráfico directo entre aplicación y proveedor y limita la exfiltración a recuentos e importes. La contabilidad en cuatro ledgers separados impide inflar el ahorro por doble cómputo, y el protocolo de doble ejecución con intervalos de confianza al 95% (31,5% en gpt-4o-mini y 75,1% en claude-sonnet-5 sobre un flujo con un 30% de repeticiones exactas, y 2,5% neto con la palanca PRO del gateway) convierte el ahorro en una magnitud medible, auditable y comparable. El trabajo futuro amplía la base empírica a más proveedores y tráfico de terceros, completa la medición A/B de la skill de suscripción y caracteriza la interacción entre ledgers cuando varias palancas se activan a la vez.

REFERENCES

- [1] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference*

- on *Empirical Methods in Natural Language Processing (EMNLP)*, 2023.
- [2] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.
 - [3] In Gim, Guojun Chen, Seung-seob Lee, Nils Sathiamoorthy, Shriram Patil, Michael Kaminsky, and Petros Maniatis. Prompt cache: Modular attention reuse for low-latency inference. *arXiv preprint arXiv:2311.04934*, 2023.
 - [4] Bin Gao, Ziming He, Purushotham Sharma, Qingyang Kang, Djordje Jia, et al. Cost-efficient large language model serving for multi-turn conversations with cached attention. In *Proceedings of the 2024 USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2024.
 - [5] Anthropic. Prompt caching — documentation. <https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching>, 2025.
 - [6] OpenAI. Prompt caching — api guide. <https://platform.openai.com/docs/guides/prompt-caching>, 2024.
 - [7] Fu Bang. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS)*, 2023.
 - [8] Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.
 - [9] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023.
 - [10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, 2017.
 - [11] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.
 - [12] Joint Committee for Guides in Metrology. Evaluation of measurement data — guide to the expression of uncertainty in measurement. Technical Report JCGM 100:2008, BIPM, 2008.
 - [13] Anthropic. Claude Code — agentic coding tool documentation. <https://www.anthropic.com/claude-code>, 2025.
 - [14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Schneider, Heiko Schulman, Dario Amodei, and Wojciech Zaremba. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
 - [15] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, 1994.