# How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality

Technical Report · Version 1.1 · October 2026

Aridio Silva · Independent Researcher · ORCID 0009-0008-2411-6995

https://doi.org/10.5281/zenodo.23732760

**Article DOI:** 10.5281/zenodo.23732760

Technical report · Critical review and extension

**SGAEIA Research Series — Article 23**

# How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality

*An evidence-based review of six levers, their limits, and a governed reference architecture*

**Aridio Silva** Independent Researcher, Brazil Creator of SGAEIA — Secure Governed Autonomous Edge Intelligence Architecture **ORCID:** 0009-0008-2411-6995

© Aridio Silva @aridiosilva · October 2026 · Version 1.1 · Text and figures licensed CC BY 4.0

## Abstract

On 26 October 2026 R. Bhardwaj published an infographic claiming that a well-designed GenAI and agentic AI stack can cut monthly LLM spend by 70%, from US$10,000 to US$2,926, while keeping answer quality the same or better [9]. This report breaks that proposal into its parts: a central thesis, six cost levers, a reference architecture, a cost decomposition and a before-and-after quality table. It then checks each part against primary sources: peer-reviewed papers, arXiv preprints and the current API documentation of the three largest model providers. The literature supports every lever, often with larger effects than the post assumes. Cascades and routers matched the strongest model's accuracy at a fraction of its cost in [10, 13, 39, 51]. Providers charge 50% less for batch work [4, 18, 40] and as little as 0.1× the input price for cached prompt reads [5]. Three weaknesses stand out. The 70% figure is illustrative, not measured. It leaves out the overhead the levers add (embeddings, routers, judges, evaluation). And the quality table reports differences smaller than the sampling error of a typical evaluation set [35]. The report adds a formal cost model, break-even conditions for cascades and caches, a sensitivity analysis yielding an illustrative 61–68% net-saving scenario under stated assumptions, without presenting it as an empirical estimate, a KPI tree centred on cost per successful task [14], a phased implementation roadmap, a glossary and an audited bibliography of 56 sources whose links were checked on 27 October 2026.

**Keywords:** LLMOps; inference cost; model routing; LLM cascades; retrieval-augmented generation; prompt caching; semantic caching; quality gates; cost-of-pass; AI governance.

## Contents

1. Introduction
2. Method
3. The central thesis
4. A cost model for LLM systems
5. The six levers
6. Reference architecture
7. Context optimisation engine
8. Caching: prompt, semantic and KV
9. Model routing and cascades
10. Quality gates and escalation
11. Cost decomposition and sensitivity
12. Evaluation and KPIs
13. Observability and governance
14. Relationship to SGAEIA

15. Implementation roadmap
16. Discussion and threats to validity
17. Conclusion
18. Glossary
19. References (audited)

# 1 Introduction

## 1.1 Context

LLM inference is now a recurring operating cost, not a research expense. Each request is billed per token, and per-token prices across models differed by up to two orders of magnitude when FrugalGPT surveyed them [10]. Agentic systems make this worse. One user request can trigger many model calls, tool calls and retries [2, 50]. Teams therefore face an engineering problem: deliver the quality the task needs at the lowest cost per successful outcome.

## 1.2 Object of analysis

The object is a LinkedIn post and infographic by Rajeev Bhardwaj [9]. Its text reads: cut LLM costs "by up to 70%, not by compromising intelligence", through smarter model routing, context, caching and agentic workflows. The infographic has six parts:

- a headline result (US$10,000 to US$2,926 per month);
- six "key levers";
- a reference architecture with six stages and a governance layer;
- an "illustrative cost breakdown" waterfall;
- a quality table comparing baseline and optimised metrics;
- a takeaway: *spend more intelligence only when the task actually needs it*.

## 1.3 Contributions

1. A structured restatement of the proposal, re-drawn as 13 original figures in English and Portuguese.
2. For each component, the primary literature that supports, qualifies or contradicts it, with the concepts explained for a non-specialist technical reader.
3. A formal cost model (Eq. 1) that maps each lever to one factor of spend, plus break-even conditions for cascades (Eq. 4) and prompt caching (Eq. 3).
4. A check of the arithmetic in the post, and a sensitivity analysis of the 70% claim that includes the overhead the levers add.
5. A KPI tree, a phased roadmap with exit gates, a glossary and an audited bibliography.

# 2 Method

The review followed four steps.

1. **Source decomposition.** Every claim in the infographic was transcribed and grouped by component: thesis, levers, architecture, decomposition, quality table, takeaway. The post text and publication date were read from the public LinkedIn page.
2. **Literature selection.** For each component we searched for (a) peer-reviewed papers or well-cited arXiv preprints that define or measure the technique, (b) the providers' own documentation for prices and limits, and (c) recognised standards for governance and reliability. Preference went to primary sources over blog summaries.
3. **Audit.** Every reference was resolved on 27 October 2026. arXiv records were checked through the arXiv API for title, authors and submission date. Every other URL was fetched and its HTTP status recorded. Numeric claims quoted in this report (for example "up to 98%" for FrugalGPT, or the 0.1× cache-read multiplier) were copied from the source abstract

or documentation page, not from secondary summaries.

4. **Synthesis and extension.** The claims were rebuilt into a cost model and tested with simple analytical and sensitivity calculations. Where the post gives no data, figures are marked *illustrative*.

Model-side efficiency techniques such as quantisation, speculative decoding and serving-system scheduling are surveyed elsewhere [33, 54] and are outside the scope of this report, which treats the model as a priced API.

**Limitations of the method.** This is a structured review, not a systematic review with a registered protocol. No experiment was run. Provider prices change often and are quoted as read on the audit date.

# 3 The central thesis

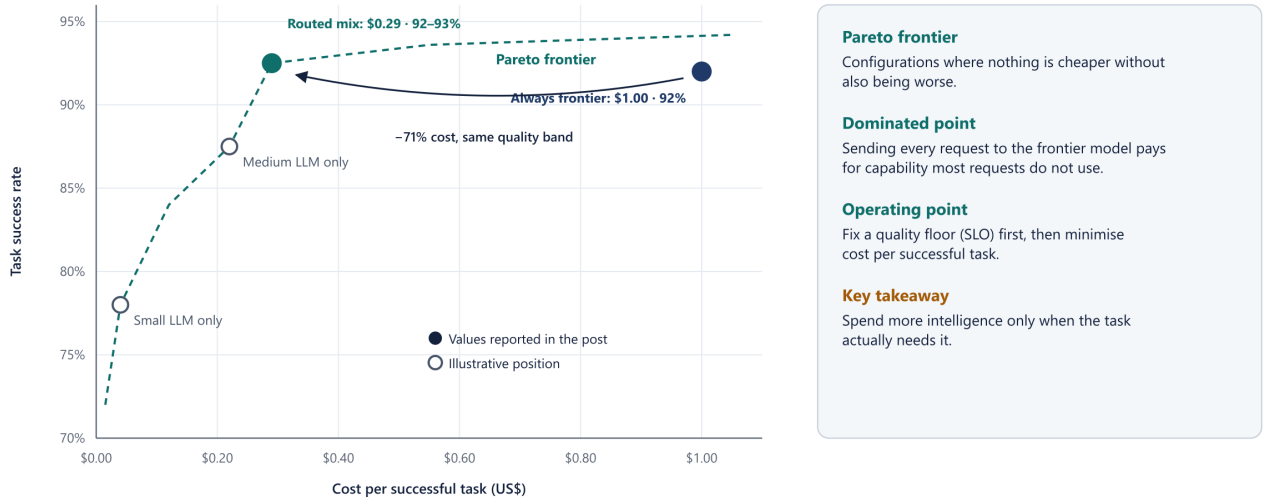
## 3.1 Statement

The thesis is that cost and quality do not have to be traded against each other across the board. Most requests in a real workload are easy. Paying frontier-model prices for them buys capability that goes unused. A system that matches capability to difficulty can therefore sit on a better point of the cost–quality curve than one that always uses the largest model.

FIGURE 1

**Central thesis: buy quality where it is needed, not everywhere**

Cost–quality plane (conceptual)




Based on R. Bhardwaj (2026), LinkedIn post, re-drawn and extended. Frontier concept: Chen et al. 2023; Kapoor et al. 2024; Erol et al. 2025.

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 1.*** *The central thesis on the cost–quality plane. The two filled points are the baseline and optimised values reported in the post; the hollow points and the frontier are illustrative.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

## 3.2 The Pareto view

A configuration is *Pareto-optimal* when no other configuration is cheaper without also being worse. Kapoor et al. showed that agent benchmarks that report accuracy alone push the field toward "needlessly complex and costly" agents, and they argue that cost and accuracy should be optimised together [25]. HELM had already argued for multi-metric evaluation that includes efficiency [28]. In this view the "always frontier" baseline in the post is a *dominated* point (Figure 1). The routed mix reaches the same success band at 29% of the cost per successful task.

## 3.3 The right unit: cost per successful task

The most important choice in the infographic is the last row of its quality table: **cost per successful task** (US$1.00 to US$0.29). Erol et al. formalise the same idea as *cost-of-pass*: the expected money spent to obtain one correct solution [14]. With per-attempt cost $c$ and success probability $p$:

CPST = $c$ / $p$

This metric stops a cheap but unreliable model from looking good. A model that costs a tenth as much but succeeds half as often is only five times cheaper per success, and that is before the cost of handling its failures.

### *Strengths of the thesis*

- Matches the empirical finding that routers and cascades keep quality at a fraction of the cost [10, 13, 39].
- Gives a single, business-readable target metric.
- Applies to agents as well as chat [25].

### *Limits*

- It depends on the workload. If most requests are hard, there is little to route away.
- "Same quality" must be shown statistically, not asserted (Section 12).
- Quality on the long tail of rare, high-stakes queries can fall while the average looks fine.

## 4 A cost model for LLM systems

The post lists levers but does not say which part of the bill each one acts on. We make that explicit with a monthly cost model:

$$C = N \cdot (1 - h) \cdot k \cdot [\, p_{in}(m) \cdot (T_u + \delta \cdot T_c) + p_{out}(m) \cdot T_{out} \,] \cdot (1 - \beta \cdot b) + C_{ovh}(1)$$

Here $N$ is requests per month, $h$ the semantic-cache hit rate, $k$ the mean number of LLM calls per request, $m$ the model chosen with its input and output prices $p$, $T_u$ and $T_c$ the uncached and cached input tokens, $\delta$ the cached-read price multiplier, $T_{out}$ the output tokens, $\beta$ the batch discount, $b$ the share of traffic sent as batch, and $C_{ovh}$ the overhead the optimisation itself adds.

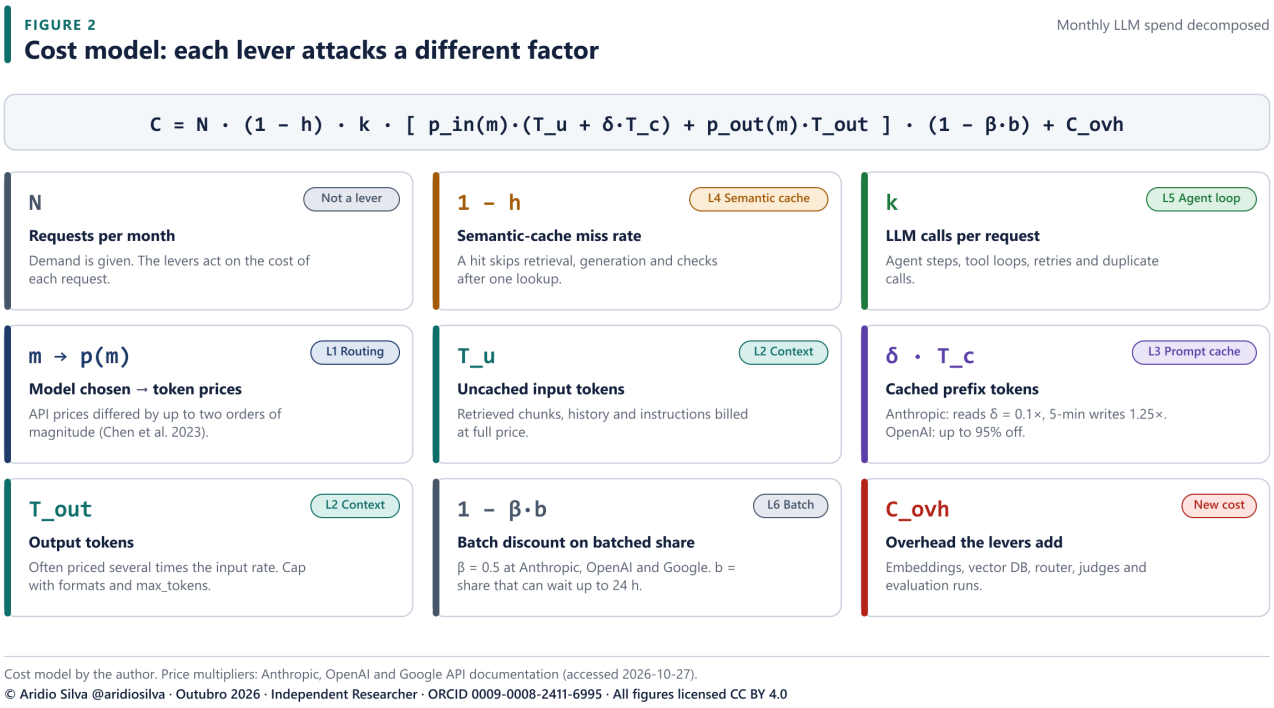



***Figure 2.*** *Each lever attacks a different factor of Eq. 1. The red card is the overhead that the levers themselves introduce and that the post leaves out.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

Two properties of Eq. 1 explain the shape of the post's waterfall. First, the factors **multiply**, so savings compound: five reductions of 15%, 40%, 25%, 15% and 10% give $1 - 0.85 \cdot 0.60 \cdot 0.75 \cdot 0.85 \cdot 0.90 = 70.7\%$, not the 105% a naive sum would suggest. Second, levers on the same factor **do not stack fully**. Once routing has moved traffic to a cheaper model, trimming its tokens saves money at the cheaper price. The order in which savings are counted changes how much each step appears to contribute, but not the final product.

Batch processing illustrates how to read the model. All three major providers discount batch work by 50% [4, 18, 40], so β = 0.5. The post's 10% batch step therefore implies *b* = 20%: one request in five can wait up to 24 hours.

# 5 The six levers

FIGURE 3

**The six levers: target, evidence, risk and trade-off**

Synthesis of the literature review

| Lever | Factor | Reported effect (source) | Risk | Effort | Main trade-off |
|---|---|---|---|---|---|
| 1 Intelligent model routing | p(m) | Up to 98% cost cut at GPT-4 accuracy (FrugalGPT); >2× cheaper (RouteLLM); 40% fewer large-model calls (Hybrid LLM). | med–high | high | Misrouted hard queries; router needs data and upkeep. |
| 2 Context engineering | T_u, T_out | Up to 20× prompt compression with little loss (LLMLingua); mid-context evidence is used worst (Liu 2023). | medium | medium | Cutting too much loses recall; too much context adds noise. |
| 3 Prompt caching | δ·T_c | Cache reads at 0.1× input price (Anthropic); up to 95% off cached input (OpenAI); 8× faster TTFT on GPU (Prompt Cache). | low | low | Needs a stable prefix; write premium; shared caches leak timing. |
| 4 Semantic caching | h | ~31% of queries were repeats in the MeanCache study; open-source GPTCache. | high | medium | False hits, stale answers and cross-tenant leaks. |
| 5 Agentic loop optimisation | k | Simple baselines matched costly SOTA agents at a fraction of the cost (Kapoor 2024). | low–med | medium | Fewer steps and retries vs. robustness on hard tasks. |
| 6 Batch processing | β·b | 50% discount, results within 24 h (Anthropic, OpenAI, Google); stacks with caching (Anthropic). | low | low | Only for work that can wait; adds queueing. |

Sources: Chen 2023; Ong 2024; Ding 2024; Jiang 2023; Gill 2024; Kapoor 2024; Anthropic, OpenAI and Google docs (2026).

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 3.*** *The six levers, the factor of Eq. 1 each one targets, the strongest effect reported in the literature, and the main trade-off.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

## 5.1 Lever 1: intelligent model routing

**Concept.** A router sends each request to the cheapest model expected to answer it well. A cascade is a sequential variant: try a cheap model first, check its answer, and escalate only if the check fails.

**Evidence.**

- FrugalGPT learned which combination of LLM APIs to query per request. It matched GPT-4's performance "with up to 98% cost reduction", or improved accuracy by 4% at the same cost [10].
- Hybrid LLM routed between a small and a large model by predicted difficulty and made "up to 40% fewer calls to the large model, with no drop in response quality" [13].
- RouteLLM trained routers on human preference data and cut cost "by over 2 times in certain cases". Its routers kept working when the strong and weak models were swapped at test time [30, 39].
- Answer-consistency cascades reached GPT-4-level reasoning accuracy at about 40% of its cost [51].
- AutoMix used few-shot self-verification and a POMDP router to cut cost by over 50% at comparable performance [1].

**Trade-offs.** Routers need labelled or preference data and drift as models change. A wrong "easy" verdict on a hard request hurts quality most on rare, high-stakes queries. Cascades pay twice for every escalated request (Section 9).

## 5.2 Lever 2: context engineering

**Concept.** Context engineering means deciding which tokens enter the model's context window and in what order. Anthropic describes it as finding "the smallest possible set of high-signal tokens" for the task [3].

**Evidence.**

- Models use information placed in the middle of a long context worse than information at the start or end ("lost in the middle") [29].
- Performance degrades as input length grows, even on simple tasks [11].
- LLMLingua compressed prompts "up to 20x" with little loss [23, 34]. LongLLMLingua extended this to long-context RAG [24].
- Budget-aware prompting can shorten reasoning output as well [22].

Fewer tokens therefore save money and can also *raise* quality.

**Trade-offs.** Cutting too hard loses the one passage that mattered (a recall loss). Compression adds a model call and latency. Aggressive budgets need evaluation per task type.

## 5.3 Lever 3: prompt caching

**Concept.** When many requests share the same prompt prefix (system instructions, tool schemas, few-shot examples, a long document), the provider can reuse the already-processed attention state of that prefix instead of recomputing it. Gim et al. showed the systems basis for this: reusing attention states cut time-to-first-token by 8× on GPU and 60× on CPU without changing outputs [17].

**Current provider terms (audited 27 Oct 2026).**

- **Anthropic:** cache reads at 0.1× the base input price (lower on some newer models). Cache writes at 1.25× for a 5-minute TTL or 2× for a 1-hour TTL. These multipliers stack with the batch discount [5].
- **OpenAI:** caching is on by default for supported models, with cached input "discounted up to 95%" and a minimum cacheable prefix of 1,024 tokens on recent models [41].
- **Google:** implicit caching on Gemini, with explicit caching available on some models [19].

**Trade-offs.** The prefix must be byte-identical across requests, so templates must put stable content first and variable content last. Writes cost more than ordinary input. Gu et al. found that caches shared across users in seven API providers create a timing side channel that can leak information about other users' prompts [20].

## 5.4 Lever 4: semantic caching

**Concept.** A semantic cache stores whole answers and returns one when a new query is *close in meaning* to an earlier one, measured by embedding similarity [45]. GPTCache is the reference open-source implementation [7, 55]. MeanCache reports that about 31% of queries in its setting were repeats, and warns that naive semantic caches have "unacceptable false hit-and-miss rates" [16].

**Trade-offs.** A false hit returns a confident answer to a different question. Answers go stale when sources change. A cache shared across tenants can leak data. Section 8 gives the required controls.

## 5.5 Lever 5: agentic loop optimisation

**Concept.** Agents interleave reasoning and tool use [50]. Each step is a paid call, and redundant steps, retries and hand-offs multiply $k$ in Eq. 1.

**Evidence.** Kapoor et al. found that simple baselines matched state-of-the-art agents on accuracy at much lower cost [25]. Anthropic's guidance is to start with the simplest pattern that works (a single call, then a fixed workflow) and add autonomy only when it pays for itself [2].

**Trade-offs.** Hard caps on steps and retries can stop an agent just before it would have succeeded. Caps should be set per task type from traces.

## 5.6 Lever 6: batch processing

**Concept.** Work that does not need an immediate answer is submitted asynchronously in bulk. Examples are enrichment, classification of backlogs, evaluation runs and report generation.

**Evidence.** Anthropic, OpenAI and Google all price batch requests at 50% of the synchronous rate and return results within 24 hours, often much sooner [4, 18, 40].

**Trade-offs.** It only applies to latency-tolerant work, and it adds queueing and result-handling code.

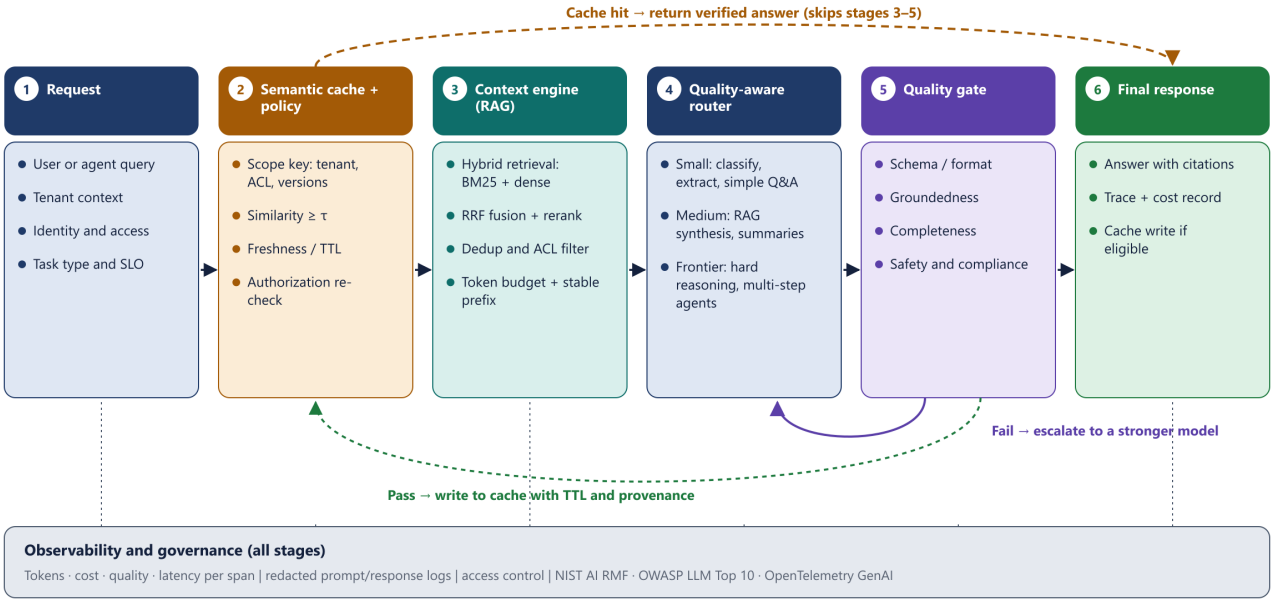
# 6 Reference architecture

The post arranges the levers into a pipeline. We re-draw it with its control loops made explicit (Figure 4).

FIGURE 4

**Reference architecture: quality-aware, cost-optimised GenAI stack**

Request path and control loops




Based on R. Bhardwaj (2026), LinkedIn post, re-drawn and extended. Governance: NIST AI 100-1, NIST AI 600-1, OWASP LLM Top 10 (2025), OpenTelemetry GenAI conventions.

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

*Figure 4. Reference architecture, re-drawn from the post. Three control loops are added explicitly: cache-hit bypass (amber), escalation (violet) and cache write-back after the gate (green).© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

1. **Request.** Carries the query, tenant, identity and access rights, and task type. Task type is the cheapest routing signal and should be captured at the edge.
2. **Semantic cache and policy check.** Returns a stored answer only if it is authorised, fresh and semantically equivalent (Section 8).
3. **Context optimisation engine.** Hybrid retrieval, rerank, filter and token budget (Section 7). This is retrieval-augmented generation as introduced by Lewis et al. [27].
4. **Quality-aware router.** Picks a small, medium or frontier model (Section 9).
5. **Quality gate.** Checks format, groundedness, completeness and safety, and escalates on failure (Section 10).
6. **Final response.** Returns the answer with citations and emits a trace with cost.

A cross-cutting **observability and governance** layer records tokens, cost, quality and latency per stage (Section 13).

Three design principles follow from the literature:

- **Cheap decisions first.** Cache lookup and deterministic checks cost far less than a model call, so they run before it.
- **Close the loop.** Only answers that passed the gate are written to the cache, which keeps errors from being replayed.
- **Prefer self-correcting retrieval.** Self-RAG [6] and Corrective RAG [49] add retrieval-quality checks inside the generation loop. They are natural extensions of stages 3 and 5.

# 7 Context optimisation engine

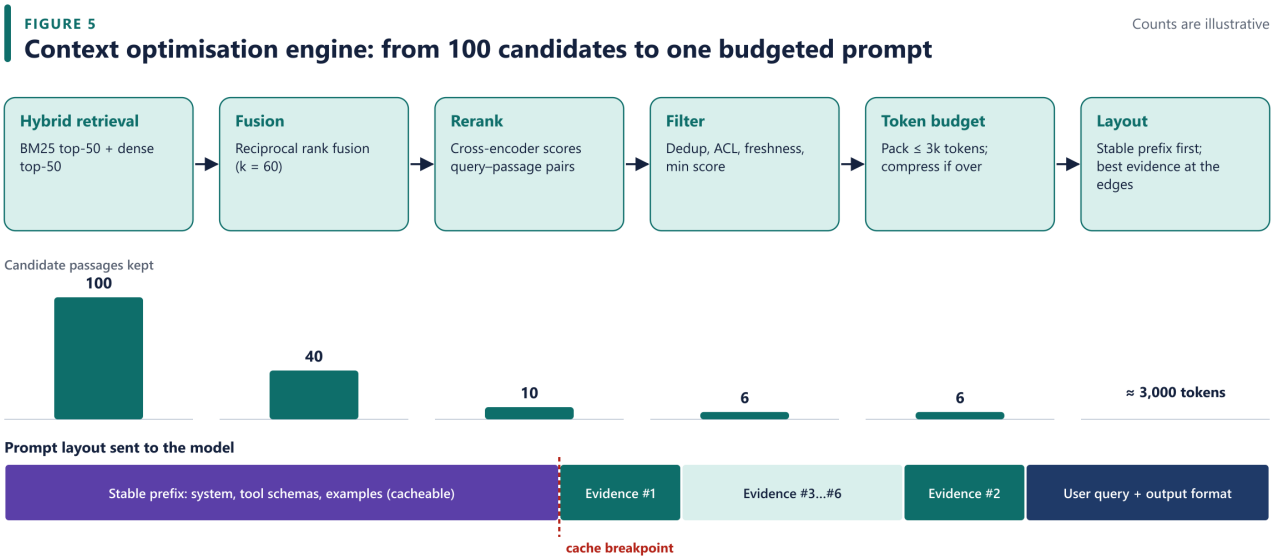



***Figure 5.*** *The context engine narrows about 100 candidate passages to a budgeted prompt. Counts are illustrative. The layout puts the cacheable prefix first and the strongest evidence at the edges.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

## 7.1 Hybrid retrieval

Lexical retrieval with BM25 scores documents by term frequency, inverse document frequency and length normalisation [46]. It handles exact identifiers, codes and rare terms well. Dense retrieval embeds queries and passages in a vector space [45] and finds paraphrases. Approximate nearest-neighbour indexes such as HNSW make it fast at scale [31]. Running both and merging the results covers the weaknesses of each.

## 7.2 Fusion with RRF

Reciprocal rank fusion merges ranked lists without calibrating their scores [12]:

RRF(*d*) = Σ*r* □ R 1 / (*k* + rank*r*(*d*)), *k* = 60(2)

## 7.3 Reranking and filtering

A cross-encoder reads the query and each candidate passage together and scores relevance much more accurately than the first-stage retriever, at higher cost per pair [38]. LLMs can also act as rerankers [48]. After reranking, filters remove near-duplicates, passages the user may not see (ACL), outdated versions and passages below a minimum score.

## 7.4 Token budgeting and layout

The packer fills a fixed budget with the best passages. If the budget is exceeded, it compresses [24] rather than truncating blindly. The layout follows two rules:

- Stable content (system prompt, tool schemas, examples) goes first so that it can be cached [5].
- The strongest evidence goes at the beginning and end of the dynamic part, where models use it best [29].

Table 1. Context-engine techniques: benefit and cost

| Technique | Main benefit | Main cost or risk |
|---|---|---|
| BM25 + dense hybrid | Recall on both exact terms and paraphrases | Two indexes to maintain |
| RRF fusion | No score calibration needed | Ignores score magnitude |
| Cross-encoder rerank | Higher precision in the top results | Latency grows with candidate count |

| Technique | Main benefit | Main cost or risk |
|---|---|---|
| Prompt compression | Up to 20× fewer tokens [23] | Extra model call; may drop details |
| Token budget | Predictable cost per request | Recall loss if set too low |

# 8 Caching: prompt, semantic and KV

The post mentions "prompt caching" and "semantic caching" as separate levers, and they are often confused with each other and with the inference engine's KV cache. Figure 6 separates them.

FIGURE 6

**Three caching layers that are often confused**

What each layer reuses and risks

| Layer | Reuses | Match rule | Saves | Main risk |
|---|---|---|---|---|
| **Semantic cache** Application layer (L4) | The whole final answer | Embedding similarity ≥ τ inside a tenant/ACL/version scope | The entire pipeline: cost and latency | False hits, stale answers, cross-tenant leakage |
| **Prompt / prefix cache** Provider API (L3) | Processed prompt prefix (attention states) | Exact prefix match; ≥ 1,024 tokens (OpenAI), explicit breakpoints (Anthropic) | Input price (reads 0.1× on Anthropic) and time to first token | Write premium, short TTL, timing side channel if shared |
| **KV cache** Inference engine (self-hosted) | Key/value tensors within and across requests | Token-level prefix tree (RadixAttention), paged blocks (PagedAttention) | GPU memory and throughput | Memory pressure and eviction policy |

Sources: Bang 2023; Gill et al. 2024; Gim et al. 2024; Kwon et al. 2023; Zheng et al. 2024; Gu et al. 2025; Anthropic and OpenAI docs (2026).

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 6.*** *Three caching layers. They reuse different things, match on different rules and carry different risks.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

The third layer matters only to teams that host their own models. Inference engines keep the attention key-value (KV) tensors of each request in GPU memory. PagedAttention manages them in fixed-size blocks to cut fragmentation [26], and RadixAttention shares them across requests through a prefix tree [53]. Provider-side prompt caching is the priced, managed form of the same idea.

## 8.1 The economics of prompt caching

Take a prefix of $P$ tokens reused by $n$ requests within the cache lifetime, at Anthropic's multipliers (write 1.25×, read 0.1×) [5]. The relative cost of the prefix is:

$$\text{cost}_{\text{cache}} / \text{cost}_{\text{no cache}} = [1.25 + 0.1 \cdot (n - 1)] / n \quad (3)$$

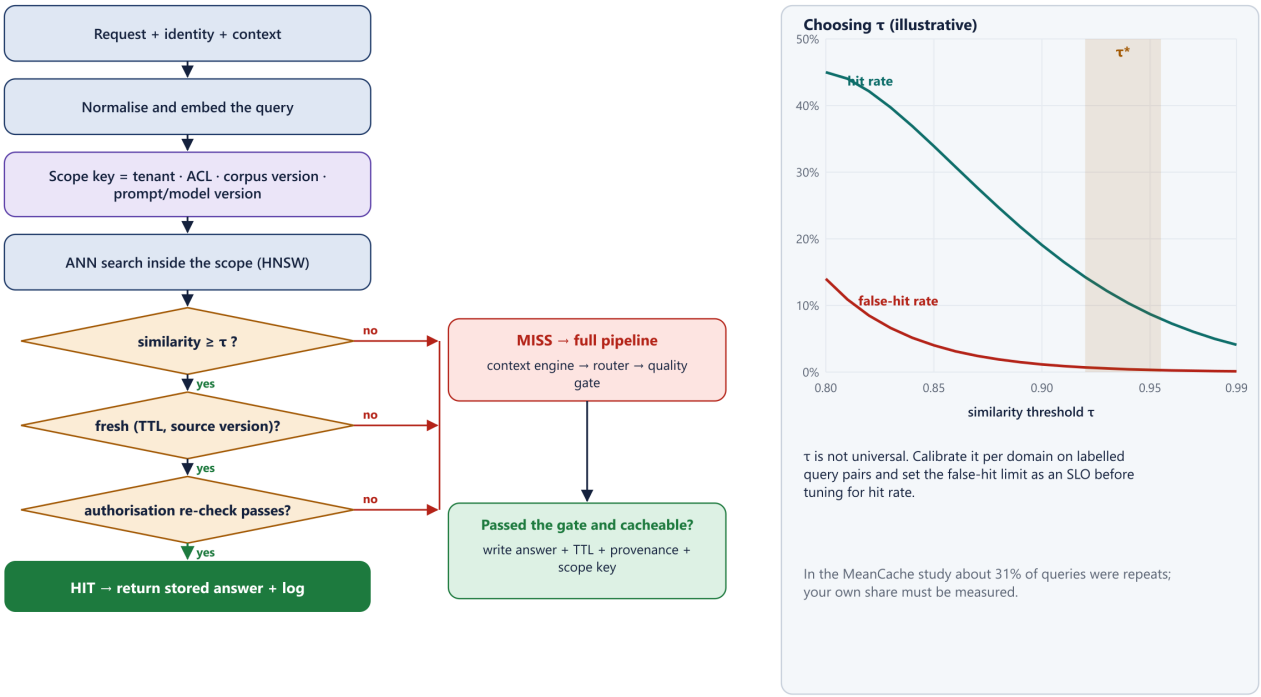
Caching pays from the second use (1.35 vs 2.0). At $n = 10$ the prefix costs 21.5% of the uncached price. The lever is low-risk and high-yield when prefixes are long and traffic is steady. It loses money only when a written prefix is never reused before it expires.

## 8.2 Semantic caching done safely

FIGURE 7

**Semantic cache: a hit must be similar, fresh and authorised**

Decision flow and threshold trade-off




Sources: Bang 2023 (GPTCache); Gill et al. 2024 (MeanCache); Malkov & Yashunin 2016 (HNSW); Reimers & Gurevych 2019. Curves are illustrative.

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

*Figure 7. Semantic-cache decision flow. A hit requires similarity, freshness and a fresh authorisation check. The threshold chart is illustrative.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

The flow in Figure 7 adds four controls that the infographic only implies ("authorized, fresh, semantically equivalent").

1. **Scope key.** Search only among entries with the same tenant, access rights, corpus version and prompt or model version. This prevents cross-tenant leakage, which OWASP lists under sensitive information disclosure and vector and embedding weaknesses [43].
2. **Calibrated threshold τ.** Raising τ cuts false hits faster than it cuts hit rate, up to a point. τ must be tuned per domain on labelled pairs [16].
3. **Freshness.** Use a TTL plus invalidation when the source document version changes.
4. **Write-back only after the gate.** Store provenance with the answer so that it can be audited or purged.

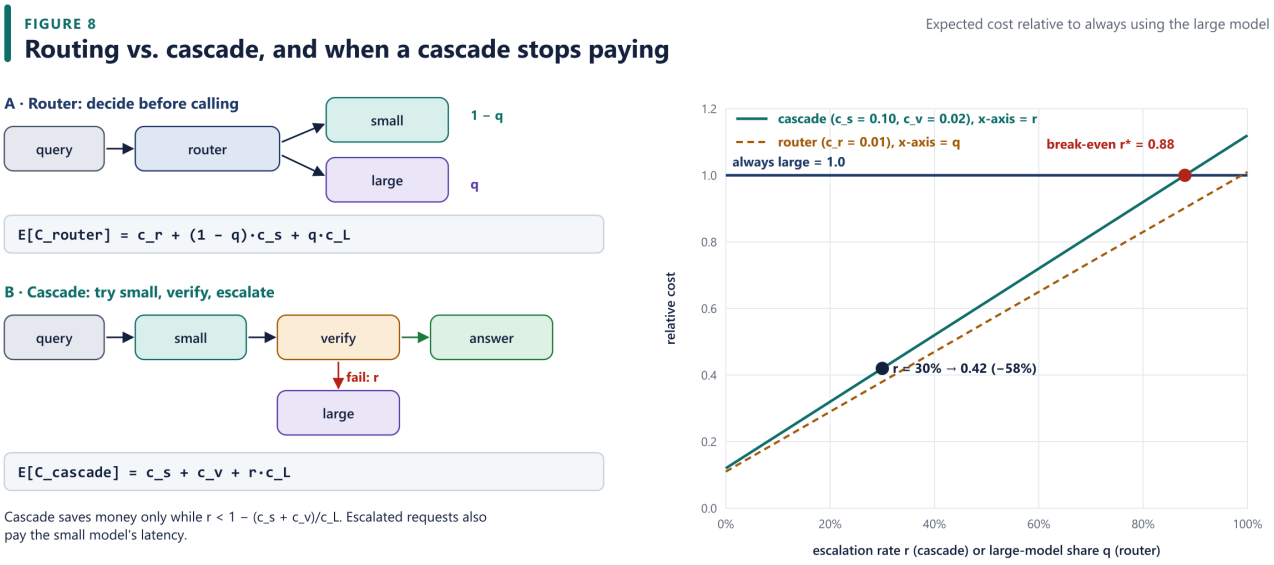
### *Benefits*

- A hit avoids the whole pipeline: the largest saving per event.
- Lower latency on repeated questions.

### *Risks*

- False hits are confident wrong answers.
- Staleness and cross-tenant leakage.
- Personalised or time-sensitive answers are poor candidates.

## 9 Model routing and cascades




Sources: Chen et al. 2023; Ding et al. 2024; Ong et al. 2024; Aggarwal et al. 2023; Yue et al. 2023; Gupta et al. 2024. Cost ratios are illustrative.
© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 8.*** *Router and cascade, their expected costs, and the break-even escalation rate. Cost ratios are illustrative.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

Let *c*s, *c*L, *c*v and *c*r be the costs of the small model, the large model, the verifier and the router. A router sends a share *q* of traffic to the large model. A cascade escalates a share *r* after verification.

E[*C*router] = *c*r + (1 − *q*)·*c*s + *q*·*c*L E[*C*cascade] = *c*s + *c*v + *r*·*c*L(4)

A cascade beats "always large" only while *r* \< 1 − (*c*s + *c*v)/*c*L. With *c*s = 0.10 and *c*v = 0.02 (relative to *c*L), the break-even is *r** = 0.88. At *r* = 30% the cascade costs 0.42, a 58% saving. That is close to the post's 40% routing step once the medium tier is included.

**Routing signals in the literature.**

- Predicted difficulty from a trained classifier [13].
- Human-preference-trained routers [39].
- Answer consistency across sampled reasoning paths [51].
- Self-verification with a POMDP policy [1].
- Token-level uncertainty, which corrects the length bias of whole-sequence confidence [21].

Table 2. Router versus cascade

| | Router (decide first) | Cascade (try, then escalate) |
|---|---|---|
| Latency | One model call plus a small router | Escalated requests pay two calls in series |
| Signal quality | Must predict difficulty before seeing an answer | Judges an actual answer, which is more reliable |
| Data needs | Labelled or preference data to train | A verifier or consistency check |
| Best for | Interactive, latency-sensitive traffic | Batch or accuracy-sensitive traffic |

## 10 Quality gates and escalation

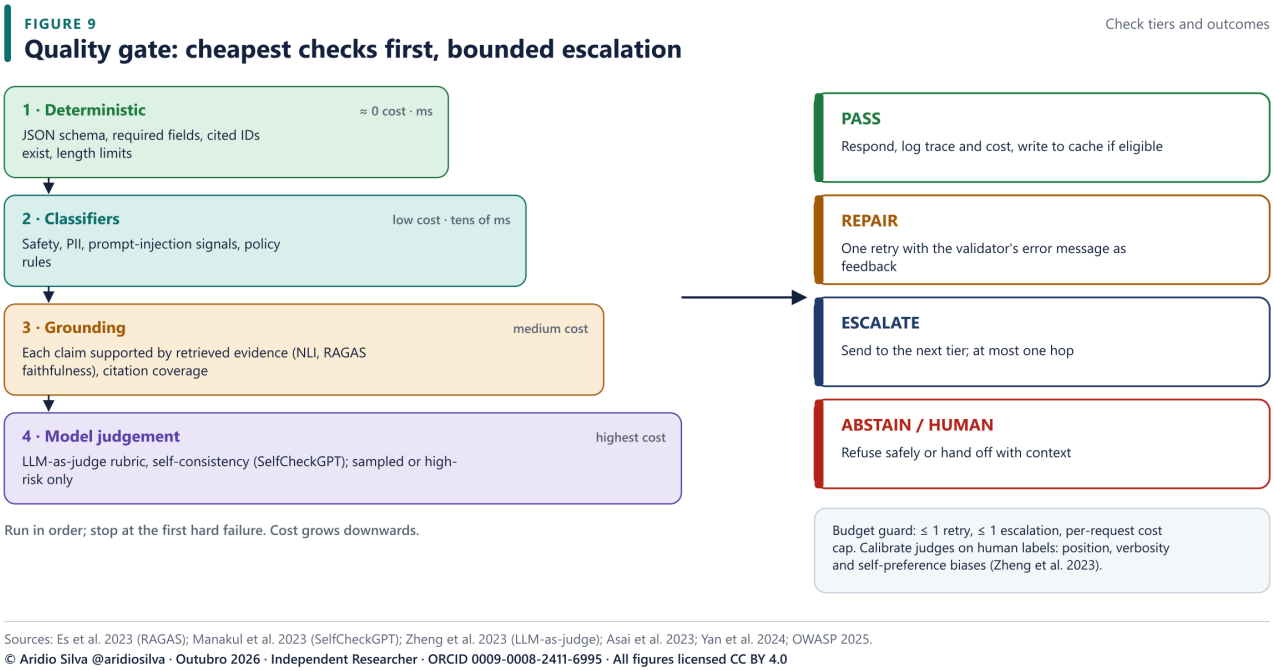



***Figure 9.*** *Quality gate: four check tiers run from cheapest to most expensive, with four possible outcomes and a budget guard.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

The gate is what makes aggressive routing safe. It should run checks in order of cost and stop at the first hard failure.

1. **Deterministic checks.** JSON schema, required fields, and whether every cited ID actually exists. These are cheap and catch the most frequent agent failures.
2. **Classifiers.** Safety, PII and prompt-injection signals (OWASP LLM01 and LLM02) [43].
3. **Grounding checks.** Is each claim supported by the retrieved evidence? RAGAS defines reference-free metrics such as faithfulness and context relevance [15, 44].
4. **Model-based judgement.** LLM-as-a-judge reaches over 80% agreement with human preferences but shows position, verbosity and self-enhancement biases [52]. SelfCheckGPT detects hallucinations by checking consistency across samples [32]. Because these checks cost as much as a generation, run them on a sample or only on high-risk routes.

**Outcomes.** Each answer ends in one of four states:

- **pass:** respond, log and possibly cache;
- **repair:** one retry with the validator's error message;
- **escalate:** send to the next model tier, at most one hop;
- **abstain or hand off:** refuse safely or pass to a human.

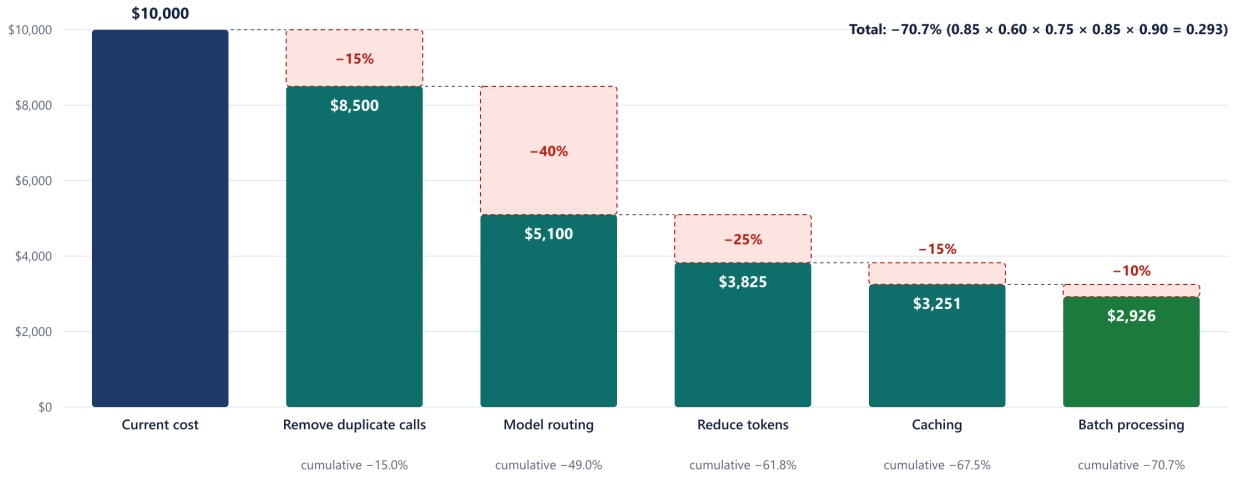
A *budget guard* caps retries, escalations and spend per request, which addresses OWASP's "unbounded consumption" risk (LLM10) [43].

## 11 Cost decomposition and sensitivity

FIGURE 10

**Cost decomposition: five sequential reductions (re-drawn to scale)**

US$ per month; values from the post




Based on R. Bhardwaj (2026), LinkedIn post, re-drawn and extended. Step percentages computed by the author; the post labels the chart as illustrative.

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 10.*** *The post's waterfall, re-drawn to scale. Step percentages and cumulative savings were computed by the author from the post's values.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

## 11.1 Checking the arithmetic

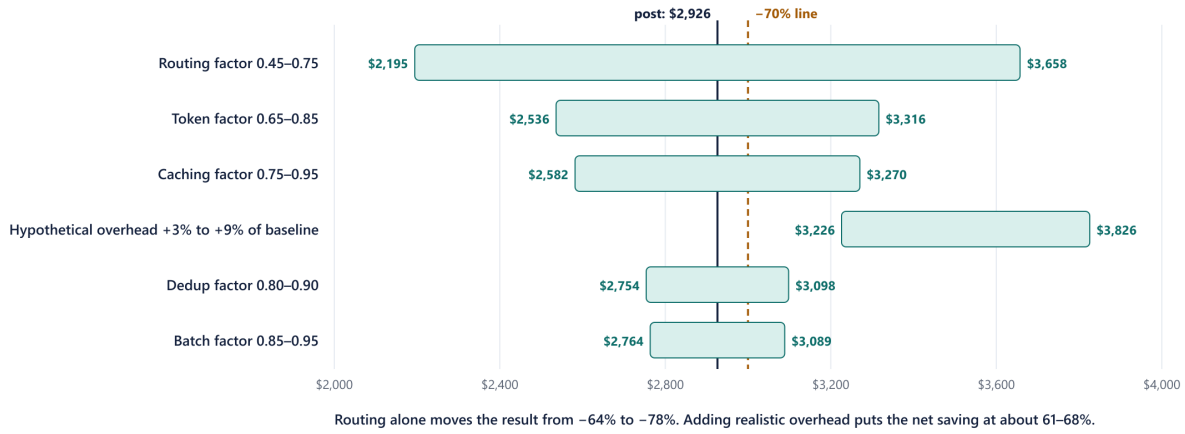
The post's values are internally consistent: 10,000 × 0.85 × 0.60 × 0.75 × 0.85 × 0.90 = 2,926. Routing is the largest single step (−40% of what remains; 34 of the 70.7 percentage points). The post labels the chart *illustrative*, so these are design targets, not measurements.

## 11.2 Sensitivity

FIGURE 11

**Sensitivity: how robust is the 70% claim?**

Final monthly cost, varying one assumption at a time




Model and ranges by the author (one-at-a-time sensitivity around the post's step factors). Illustrative, not measured.

© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 11.*** *One-at-a-time sensitivity of the final monthly cost. Ranges are the author's assumptions around the post's step factors. The five teal rows vary those factors up and down; the amber row is not a variation but the overhead the levers add, so it only increases cost.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

Varying each factor within a hypothetical author-selected range shows that, in this scenario, routing dominates the outcome (−64% to −78%). Batch and deduplication matter least. The post leaves out overhead: embeddings and the vector store for the semantic cache, the router, judge calls, and evaluation runs. If that overhead is assumed to add

3–9% of the original baseline, the **scenario's net saving falls to about 61–68%**. This calculation demonstrates sensitivity rather than an expected range for real workloads; that range must be measured per workload.

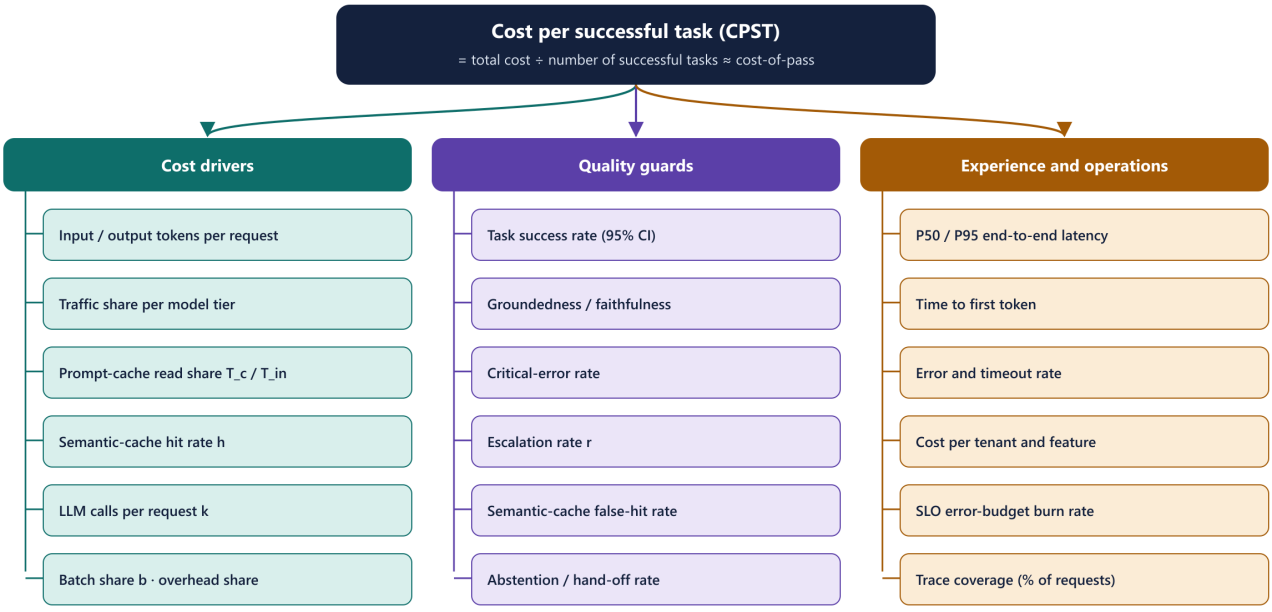
Table 3. Costs the levers add (not shown in the post)

| Item | Introduced by | How to keep it small |
|---|---|---|
| Query embeddings, vector search | L4, L2 | Small embedding model; cache embeddings |
| Router inference | L1 | Lightweight classifier, not an LLM |
| Verifier and judge calls | Gate | Deterministic checks first; sample model judges |
| Cache writes (1.25×–2×) | L3 | Cache only prefixes that will be reused |
| Evaluation runs and labelling | All | Run evaluations through the batch API (−50%) |
| Engineering and operations time | All | Phase the work; quick wins first (Section 15) |

# 12 Evaluation and KPIs

FIGURE 12
**KPI tree: one north-star metric, three families of drivers**

What to measure before and after




Sources: Erol et al. 2025 (cost-of-pass); Miller 2024 (error bars); Liang et al. 2022 (HELM); Beyer et al. 2016 (SLOs); OpenTelemetry GenAI.
© Aridio Silva @aridiosilva · Outubro 2026 · Independent Researcher · ORCID 0009-0008-2411-6995 · All figures licensed CC BY 4.0

***Figure 12.*** *KPI tree. Cost per successful task is the north-star metric; cost drivers, quality guards and operational metrics explain its movement.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

## 12.1 Reading the quality table critically

The post reports task success 92% → 92–93%, groundedness 90% → 90–91% and critical errors 0.5% → 0.4%. Miller recommends treating evaluation items as a sample from a larger population and reporting confidence intervals [35]. With 500 items, the standard error of a 92% success rate is $\sqrt{(0.92 \cdot 0.08 / 500)} \approx 1.2$ points, so the 95% interval is roughly ±2.4 points. The reported changes are inside that noise. "Same quality" is plausible but not shown.

A rigorous version would:

- evaluate both systems on the *same* items (a paired design);
- pre-register a non-inferiority margin, for example "success no more than 1 point lower";

- size the evaluation set so the interval is narrower than that margin.

### 12.2 KPI definitions

Table 4. Recommended KPIs

| KPI | Definition | Why it matters |
|---|---|---|
| Cost per successful task | Total cost ÷ successful tasks [14] | North-star; stops "cheap but wrong" from looking good |
| Tokens per request (in/out) | Mean and P95, per route | Shows context bloat and verbose outputs |
| Model-tier mix | Share of requests per tier | Main driver of p(m) |
| Prompt-cache read share | Cached ÷ total input tokens | Verifies that Lever 3 works |
| Semantic-cache hit and false-hit rates | Hits ÷ lookups; wrong hits ÷ audited hits | Saving and safety of Lever 4 |
| Escalation rate r | Escalated ÷ gated requests | Cascade break-even (Eq. 4) |
| Task success, groundedness, critical errors | On a fixed eval set, with 95% CIs [35] | Quality guards |
| P50/P95 latency and TTFT | End to end and per stage | User experience; latency SLOs [8] |
| Calls per request k | LLM calls ÷ requests | Agent-loop efficiency (Lever 5) |

## 13 Observability and governance

None of the levers can be tuned without per-request telemetry. The OpenTelemetry semantic conventions for generative AI define standard span attributes for model, token usage and operations [42]. Using them makes cost per span portable across vendors.

Governance has three anchors:

- the NIST AI Risk Management Framework, with its Govern, Map, Measure and Manage functions [36];
- the NIST Generative AI Profile [37];
- the OWASP Top 10 for LLM Applications 2025, which names the main risks this architecture must control: prompt injection, sensitive information disclosure, vector and embedding weaknesses, and unbounded consumption [43].

Sculley et al.'s warning about hidden technical debt in ML systems applies directly [47]. Every cache, router and judge is another component whose configuration can drift. Each one needs an owner, a metric and an alert.

## 14 Relationship to SGAEIA: efficiency under governance and evidence

**Classification of this section:** SGAEIA INTERPRETATION — a public conceptual connection, not normative architecture or implementation evidence.

For SGAEIA — Secure Governed Autonomous Edge Intelligence Architecture — cost reduction is not a standalone objective. An optimisation is acceptable only when it preserves authority boundaries, policy enforcement, traceability, revocation, tenant separation, and enough evidence to explain why each request followed a particular route. Cost per successful task must therefore be paired with critical-error rates, gate false positives and false negatives, per-route latency, trace completeness, and residual risk; a cheaper answer that weakens those controls is not a net gain for a governed autonomous system [56].

The six levers reviewed here relate most directly to four SGAEIA responsibility planes. Routing and cascades belong to integration and substitution: changing the model must not silently expand available authority or invalidate security assumptions. Caches and the context engine cross identity and trust as well as authority and delegation, because a reused answer must respect the current tenant, current authorisation, provenance, and corpus version. Step, retry, and spend limits support governed execution, while per-request telemetry, route decisions, and gate outcomes support observability and evidence.

This connection yields a candidate property that is not yet normative: **cost optimisation in an AI system should not degrade authority boundaries, policy enforcement, revocation capability, or the completeness of evidence required for audit**. The six-lever architecture in this report is external evidence and a possible engineering pattern; it does not become part of SGAEIA merely by appearing in this publication. Any adoption would require an existing-coverage audit, impact analysis, the applicable Research-to-Architecture gates, acceptance criteria, and adversarial verification before a decision by the project owner.

This reading also changes the practical ordering of the levers. Instrumentation, duplicate-call removal, loop limits, and prompt caching offer gains with a smaller risk surface. Routing, model judges, and semantic caching require non-inferiority evaluation, leakage tests, false-hit analysis, model-substitution checks, and safe failure modes. The objective is not the lowest possible bill, but the lowest cost compatible with governed execution and continuous, evidence-backed assurance.

# 15 Implementation roadmap

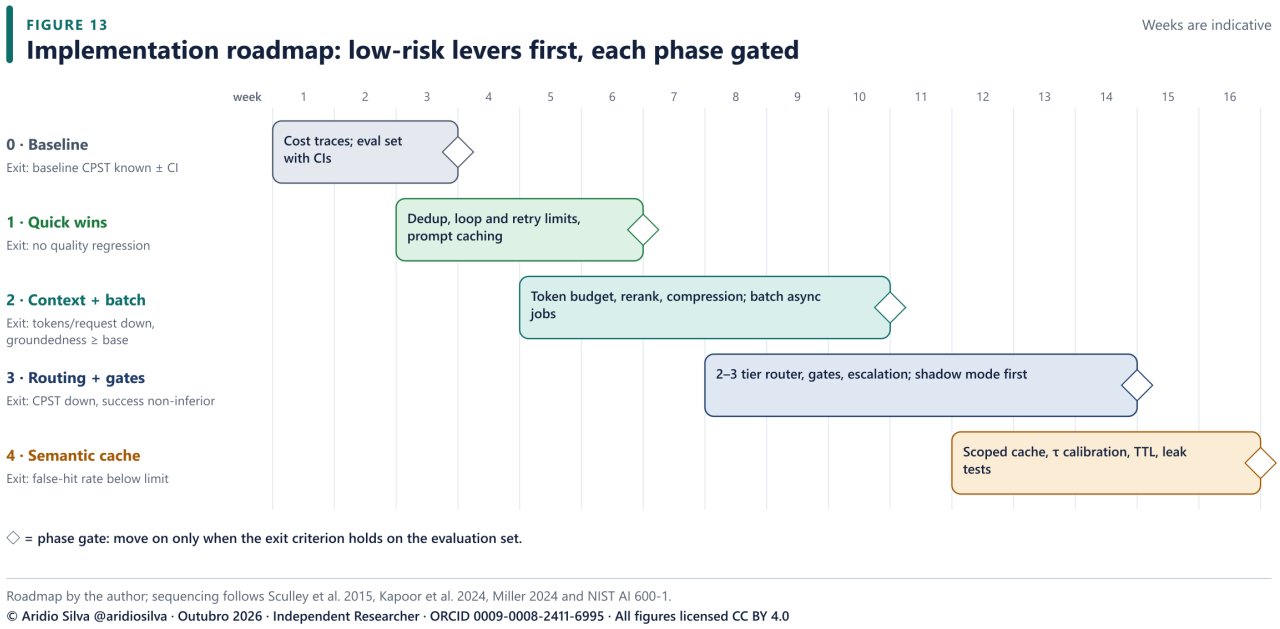



***Figure 13.*** *Phased roadmap. Each phase ends at a gate whose exit criterion must hold on the evaluation set before the next phase starts. Weeks are indicative.© 2026 Aridio Silva (@aridiosilva), Independent Researcher, ORCID 0009-0008-2411-6995. CC BY 4.0.*

Table 5. Phases, actions and supporting sources

| Phase | Actions | Exit criterion | Support |
|---|---|---|---|
| 0 · Baseline | Trace every call with tokens and cost; build an evaluation set with confidence intervals | Baseline cost per successful task known ± CI | [14, 35, 42] |
| 1 · Quick wins | Remove duplicate calls; cap loops and retries; restructure prompts for prefix caching | No quality regression; cache read share at target | [2, 5, 25, 41] |
| 2 · Context and batch | Hybrid retrieval, rerank, token budget, compression; move async jobs to batch | Tokens per request down; groundedness ≥ baseline | [12, 24, 29, 38, 40] |
| 3 · Routing and gates | 2–3 tier router; gate tiers; bounded escalation; run in shadow mode before switching | Cost per successful task down; success non-inferior | [13, 15, 39, 51, 52] |
| 4 · Semantic cache | Scoped cache; τ calibration on labelled pairs; TTL and invalidation; leakage tests | False-hit rate below the agreed limit | [7, 16, 43] |

The order reverses the post's visual order on purpose: the lowest-risk, highest-certainty levers come first. Semantic caching comes last because it is the only lever that can return a wrong answer with no model involved, so it needs the most mature governance.

## 16 Discussion and threats to validity

- **Workload dependence.** Every number in the post depends on the request mix. A workload of hard, unique, multi-step tasks has little to route or cache.
- **Price volatility.** Provider prices and cache multipliers change several times a year, and the gap between model tiers can shrink [14]. The architecture stays valid; the size of the saving does not.
- **Latency claim.** The post reports a lower P95 latency (2.8 s → 2.6 s). That is plausible if cache hits and smaller models outweigh the extra stages, but cascades add serial calls for escalated requests. It must be measured per route.
- **Evaluation validity.** Agent benchmarks often lack proper hold-out sets [25]. Routers tuned on the evaluation set will overstate gains.
- **Security of caches.** Both cache types create cross-user information channels if shared [20, 43].
- **Scope of this report.** This is a literature-grounded analysis, not an experiment. Its numeric extensions (Figures 8 and 11) are illustrative models.

## 17 Conclusion

Bhardwaj's proposal is technically sound and matches the published evidence. Every one of its six levers has primary-source support, and several published results exceed the reductions it assumes. Its main contribution is a principle: spend intelligence where the task needs it, and measure the result as cost per successful task. Its main weakness is the headline number. It is illustrative, it omits the overhead the levers add, and its quality comparison lacks statistical support.

Adding those pieces makes the modelled scenario yield a 61–68% net saving under the selected assumptions, but it does not establish an expected production range. Demonstrating a reduction near 70% requires workload telemetry, complete costs, and a paired non-inferiority test. The safest route is the one in Section 15: measure first, take the low-risk levers, and add routing and semantic caching only behind quality gates and a non-inferiority test.

## Glossary

**Agentic loop** - The cycle in which an agent reasons, calls a tool, reads the result and decides the next step. Each step is a paid model call.

**ANN (approximate nearest neighbour)** - Search that finds vectors close to a query vector quickly, at the price of occasionally missing the exact nearest one.

**Batch API** - An asynchronous endpoint that processes many requests within a time window (24 h) at a discount (50% at the major providers).

**BM25** - A lexical ranking function based on term frequency, inverse document frequency and document length.

**Cascade** - A sequence of models from cheap to expensive. A request moves to the next model only when a check on the current answer fails.

**Confidence interval (CI)** - A range that, under repeated sampling, contains the true value at the stated rate (for example 95%).

**Context engineering** - Choosing which tokens enter the model's context window, and in what order, for a given step.

**Context window** - The maximum number of tokens a model can read in one call, including instructions, evidence and history.

**Cost-of-pass / cost per successful task** - Expected spend to obtain one correct result: cost per attempt divided by success probability.

**Cross-encoder** - A model that reads query and passage together to score their relevance. More accurate and slower than comparing separate embeddings.

**Deferral rule** - The rule a cascade uses to decide whether to keep an answer or pass the request to a stronger model.

**Dense retrieval / embedding** - Representing text as vectors so that texts with similar meaning are close together, then searching by vector similarity.

**Escalation** - Re-sending a request to a more capable model after a quality check fails.

**False hit** - A semantic-cache hit that returns an answer to a different question.

**Frontier model** - The most capable and usually most expensive model tier available at a given time.

**Groundedness / faithfulness** - The degree to which every claim in an answer is supported by the evidence given to the model.

**Hallucination** - Fluent output that is not supported by the input or by facts.

**HNSW** - Hierarchical navigable small world graph: a widely used index for fast approximate nearest-neighbour search.

**Hybrid retrieval** - Running lexical (BM25) and dense retrieval together and merging their results.

**KV cache** - The attention key and value tensors an inference engine keeps in memory so that earlier tokens are not recomputed.

**LLM-as-a-judge** - Using a language model to grade another model's output against a rubric.

**Lost in the middle** - The finding that models use information in the middle of a long context worse than information at its start or end.

**Model routing** - Choosing, per request, which model should answer it, before any model is called.

**Non-inferiority test** - A test that shows a new system is not worse than the old one by more than a pre-set margin.

**Observability / trace / span** - Recording each request as a trace made of spans (one per stage), with attributes such as tokens, cost and latency.

**Pareto frontier** - The set of options where no option is better on one objective without being worse on another.

**Prompt (prefix) caching** - Provider-side reuse of the processed state of a prompt prefix that repeats across requests, billed at a reduced rate.

**Prompt compression** - Removing low-information tokens from a prompt with a smaller model while keeping its meaning.

**Prompt injection** - Input crafted to override a model's instructions. OWASP's top risk for LLM applications.

**RAG (retrieval-augmented generation)** - Retrieving relevant documents and adding them to the prompt so that the model answers from them.

**Reranking** - A second, more accurate scoring pass over the top candidates from a first-stage retriever.

**RRF (reciprocal rank fusion)** - Merging ranked lists by summing 1/(k + rank) for each document.

**Semantic cache** - An application-level cache that returns a stored answer when a new query is similar enough in meaning to a past one.

**Shadow mode** - Running a new component on live traffic without using its output, to compare it with the current one.

**SLO / error budget** - A target for a service metric, and the amount of failure allowed before action is required.

**Token** - The unit of text a model reads and writes, and the unit of billing. Roughly ¾ of an English word.

**Token budget** - A fixed cap on the tokens a stage may send or generate.

**TTFT (time to first token)** - Latency from sending a request to receiving the first output token.

**TTL (time to live)** - How long a cache entry stays valid before it expires.

## References (audited)

All links were resolved on 27 October 2026. arXiv records were checked against the arXiv API (title, authors, date). Other links were fetched and their HTTP status recorded. Where a publisher blocks automated access (HTTP 403), the DOI was confirmed to resolve to the publisher's record. Venues are given only where confirmed; otherwise the arXiv version is cited.


1. Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., et al. (2023). *AutoMix: Automatically mixing language models*. arXiv preprint arXiv:2310.12963. https://arxiv.org/abs/2310.12963Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

2. Anthropic (2024, December 19). *Building effective AI agents*. Anthropic Engineering. https://www.anthropic.com/research/building-effective-agentsVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

3. Anthropic (2025, September 29). *Effective context engineering for AI agents*. Anthropic Engineering. https://www.anthropic.com/research/effective-context-engineering-for-ai-agentsVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

4. Anthropic (2026). *Batch processing*. Claude Developer Platform documentation. https://platform.claude.com/docs/en/build-with-claude/batch-processingVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

5. Anthropic (2026). *Prompt caching*. Claude Developer Platform documentation. https://platform.claude.com/docs/en/build-with-claude/prompt-cachingVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

6. Asai, A., Wu, Z., Wang, Y., Sil, A., & Hajishirzi, H. (2023). *Self-RAG: Learning to retrieve, generate, and critique through self-reflection*. ICLR 2024. arXiv:2310.11511. https://arxiv.org/abs/2310.11511Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

7. Bang, F. (2023). *GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings*. Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023), ACL. https://aclanthology.org/2023.nlposs-1.24/Verified: HTTP 200, title checked · Accessed 2026-10-27

8. Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.) (2016). *Site reliability engineering: How Google runs production systems. Chapter 4: Service level objectives*. O'Reilly Media. https://sre.google/sre-book/service-level-objectives/Verified: HTTP 200, title checked · Accessed 2026-10-27

9. Bhardwaj, R. (2026, October 26). *Cut LLM costs by up to 70%, not by compromising intelligence, but by engineering smarter model routing, context, caching, and agentic workflows [Post with infographic]*. LinkedIn. https://www.linkedin.com/posts/rajeev-bhardwaj_genai-agenticai-llmoptimization-share-7520411441674387456-qL-QVerified: HTTP 200, title checked · Accessed 2026-10-27

10. Chen, L., Zaharia, M., & Zou, J. (2023). *FrugalGPT: How to use large language models while reducing cost and improving performance*. arXiv preprint arXiv:2305.05176. https://arxiv.org/abs/2305.05176Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

11. Chroma Research (2025). *Context rot: How increasing input tokens impacts LLM performance*. Chroma technical report. https://www.trychroma.com/research/context-rotVerified: HTTP 200 at the current URL shown · Accessed

12. Cormack, G. V., Clarke, C. L. A., & Büttcher, S. (2009). *Reciprocal rank fusion outperforms Condorcet and individual rank learning methods*. Proceedings of the 32nd ACM SIGIR Conference, 758–759. https://doi.org/10.1145/1571941.1572114Verified: DOI resolves; publisher blocks automated access (HTTP 403) · Accessed 2026-10-27

13. Ding, D., Mallick, A., Wang, C., Sim, R., et al. (2024). *Hybrid LLM: Cost-efficient and quality-aware query routing*. ICLR 2024. arXiv:2404.14618. https://arxiv.org/abs/2404.14618Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

14. Erol, M. H., El, B., Suzgun, M., Yuksekgonul, M., et al. (2025). *Cost-of-pass: An economic framework for evaluating language models*. arXiv preprint arXiv:2504.13359. https://arxiv.org/abs/2504.13359Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

15. Es, S., James, J., Espinosa-Anke, L., & Schockaert, S. (2023). *RAGAS: Automated evaluation of retrieval augmented generation*. EACL 2024 System Demonstrations. arXiv:2309.15217. https://arxiv.org/abs/2309.15217Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

16. Gill, W., Elidrisi, M., Kalapatapu, P., Ahmed, A., et al. (2024). *MeanCache: User-centric semantic caching for LLM web services*. arXiv preprint arXiv:2403.02694. https://arxiv.org/abs/2403.02694Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

17. Gim, I., Chen, G., Lee, S., Sarda, N., et al. (2023). *Prompt cache: Modular attention reuse for low-latency inference*. MLSys 2024. arXiv:2311.04934. https://arxiv.org/abs/2311.04934Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

18. Google (2026). *Batch API*. Gemini API documentation. https://ai.google.dev/gemini-api/docs/batch-apiVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

19. Google (2026). *Context caching*. Gemini API documentation. https://ai.google.dev/gemini-api/docs/cachingVerified: HTTP 200, title checked · Accessed 2026-10-27

20. Gu, C., Li, X. L., Kuditipudi, R., Liang, P., & Hashimoto, T. (2025). *Auditing prompt caching in language model APIs*. arXiv preprint arXiv:2502.07776. https://arxiv.org/abs/2502.07776Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

21. Gupta, N., Narasimhan, H., Jitkrittum, W., Rawat, A. S., et al. (2024). *Language model cascades: Token-level uncertainty and beyond*. arXiv preprint arXiv:2404.10136. https://arxiv.org/abs/2404.10136Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

22. Han, T., Wang, Z., Fang, C., Zhao, S., et al. (2024). *Token-budget-aware LLM reasoning*. arXiv preprint arXiv:2412.18547. https://arxiv.org/abs/2412.18547Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

23. Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., & Qiu, L. (2023). *LLMLingua: Compressing prompts for accelerated inference of large language models*. EMNLP 2023. arXiv:2310.05736. https://arxiv.org/abs/2310.05736Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

24. Jiang, H., Wu, Q., Luo, X., Li, D., et al. (2023). *LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression*. ACL 2024. arXiv:2310.06839. https://arxiv.org/abs/2310.06839Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

25. Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., & Narayanan, A. (2024). *AI agents that matter*. arXiv preprint arXiv:2407.01502. https://arxiv.org/abs/2407.01502Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

26. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., et al. (2023). *Efficient memory management for large language model serving with PagedAttention*. SOSP 2023. arXiv:2309.06180. https://arxiv.org/abs/2309.06180Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

27. Lewis, P., Perez, E., Piktus, A., Petroni, F., et al. (2020). *Retrieval-augmented generation for knowledge-intensive NLP tasks*. NeurIPS 2020. arXiv:2005.11401. https://arxiv.org/abs/2005.11401Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

28. Liang, P., Bommasani, R., Lee, T., Tsipras, D., et al. (2022). *Holistic evaluation of language models*. Transactions on Machine Learning Research. arXiv:2211.09110. https://arxiv.org/abs/2211.09110Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

29. Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., et al. (2024). *Lost in the middle: How language models use long contexts*. Transactions of the ACL, 12. https://doi.org/10.1162/tacl*a*00638. https://arxiv.org/abs/2307.03172Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

30. LMSYS (2024–2026). *RouteLLM: A framework for serving and evaluating LLM routers [Software]*. GitHub. https://github.com/lm-sys/RouteLLMVerified: HTTP 200, title checked · Accessed 2026-10-27

31. Malkov, Y. A., & Yashunin, D. A. (2016). *Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs*. IEEE TPAMI (2020). arXiv:1603.09320. https://arxiv.org/abs/1603.09320Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

32. Manakul, P., Liusie, A., & Gales, M. J. F. (2023). *SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models*. EMNLP 2023. arXiv:2303.08896. https://arxiv.org/abs/2303.08896Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

33. Miao, X., Oliaro, G., Zhang, Z., Cheng, X., et al. (2023). *Towards efficient generative large language model serving: A survey from algorithms to systems*. arXiv preprint arXiv:2312.15234. https://arxiv.org/abs/2312.15234Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

34. Microsoft (2023–2026). *LLMLingua [Software]*. GitHub. https://github.com/microsoft/LLMLinguaVerified: HTTP 200, title checked · Accessed 2026-10-27

35. Miller, E. (2024). *Adding error bars to evals: A statistical approach to language model evaluations*. arXiv preprint arXiv:2411.00640. https://arxiv.org/abs/2411.00640Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

36. National Institute of Standards and Technology (2023). *Artificial intelligence risk management framework (AI RMF 1.0)*. NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1Verified: HTTP 200 at the current URL shown · Accessed 2026-10-27

37. National Institute of Standards and Technology (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile*. NIST AI 600-1. https://doi.org/10.6028/NIST.AI.600-1Verified: HTTP 200 at the current URL shown · Accessed 2026-10-27

38. Nogueira, R., & Cho, K. (2019). *Passage re-ranking with BERT*. arXiv preprint arXiv:1901.04085. https://arxiv.org/abs/1901.04085Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

39. Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., et al. (2024). *RouteLLM: Learning to route LLMs with preference data*. arXiv preprint arXiv:2406.18665. https://arxiv.org/abs/2406.18665Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

40. OpenAI (2026). *Batch API*. OpenAI API documentation. https://developers.openai.com/api/docs/guides/batchVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

41. OpenAI (2026). *Prompt caching*. OpenAI API documentation. https://developers.openai.com/api/docs/guides/prompt-cachingVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

42. OpenTelemetry (2026). *Semantic conventions for generative AI systems*. GitHub: open-telemetry/semantic-conventions-genai. https://github.com/open-telemetry/semantic-conventions-genaiVerified: HTTP 200, title checked · Accessed 2026-10-27

43. OWASP GenAI Security Project (2025). *OWASP Top 10 for LLM applications 2025*. OWASP Foundation. https://genai.owasp.org/llm-top-10/Verified: HTTP 200, title checked · Accessed 2026-10-27

44. Ragas contributors (2023–2026). *Ragas: Evaluation toolkit for LLM applications [Software]*. GitHub. https://github.com/vibrantlabsai/ragasVerified: HTTP 200 at the current URL shown · Accessed 2026-10-27

45. Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence embeddings using Siamese BERT-networks*. EMNLP-IJCNLP 2019, ACL. https://doi.org/10.18653/v1/D19-1410. https://aclanthology.org/D19-1410/Verified: HTTP 200 at the current URL shown · Accessed 2026-10-27

46. Robertson, S., & Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond*. Foundations and Trends in Information Retrieval, 3(4), 333–389. https://doi.org/10.1561/1500000019Verified: DOI resolves; publisher blocks automated access (HTTP 403) · Accessed 2026-10-27

47. Sculley, D., Holt, G., Golovin, D., Davydov, E., et al. (2015). *Hidden technical debt in machine learning systems*. Advances in Neural Information Processing Systems 28. https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.htmlVerified: HTTP 200, title checked · Accessed 2026-10-27

48. Sun, W., Yan, L., Ma, X., Wang, S., et al. (2023). *Is ChatGPT good at search? Investigating large language models as re-ranking agents*. EMNLP 2023. arXiv:2304.09542. https://arxiv.org/abs/2304.09542Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

49. Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). *Corrective retrieval augmented generation*. arXiv preprint arXiv:2401.15884. https://arxiv.org/abs/2401.15884Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

50. Yao, S., Zhao, J., Yu, D., Du, N., et al. (2022). *ReAct: Synergizing reasoning and acting in language models*. ICLR 2023. arXiv:2210.03629. https://arxiv.org/abs/2210.03629Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

51. Yue, M., Zhao, J., Zhang, M., Du, L., & Yao, Z. (2023). *Large language model cascades with mixture of thoughts representations for cost-efficient reasoning*. ICLR 2024. arXiv:2310.03094. https://arxiv.org/abs/2310.03094Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

52. Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., et al. (2023). *Judging LLM-as-a-judge with MT-Bench and Chatbot Arena*. NeurIPS 2023 Datasets and Benchmarks. arXiv:2306.05685. https://arxiv.org/abs/2306.05685Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

53. Zheng, L., Yin, L., Xie, Z., Sun, C., et al. (2024). *SGLang: Efficient execution of structured language model programs*. NeurIPS 2024. arXiv:2312.07104. https://arxiv.org/abs/2312.07104Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

54. Zhou, Z., Ning, X., Hong, K., Fu, T., et al. (2024). *A survey on efficient inference for large language models*. arXiv preprint arXiv:2404.14294. https://arxiv.org/abs/2404.14294Verified: arXiv API metadata match, HTTP 200 · Accessed 2026-10-27

55. Zilliz (2023–2026). *GPTCache: Semantic cache for LLMs [Software]*. GitHub. https://github.com/zilliztech/GPTCacheVerified: HTTP 200, title checked · Accessed 2026-10-27

56. Silva, Aridio. (2026). *SGAEIA: Secure Governed Autonomous Edge Intelligence Architecture* (Version 0.3.4) [Research software artifact]. Zenodo. https://doi.org/10.5281/zenodo.22557796

## How to cite

Silva, Aridio. (2026). *How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality: An evidence-based review of six levers, their limits, and a governed reference architecture* (Version 1.1) [Technical report]. Independent research. ORCID 0009-0008-2411-6995. DOI: https://doi.org/10.5281/zenodo.23732760.

Original analysis, figures and translation by Aridio Silva (@aridiosilva). The proposal under review belongs to its author, R. Bhardwaj [9]; the figures re-draw and extend it with attribution.

## About the Author

**Aridio Silva** is an independent researcher based in Brazil working on the architecture, security, governance, and trustworthiness of autonomous and distributed artificial intelligence systems.

His research focuses on **Agentic AI, Multi-Agent Systems, Edge AI, AI Security, Zero Trust, Security-by-Design, AI Governance, Spec-Driven Development, and continuous security assurance**.

He is the creator and lead researcher of **SGAEIA — Secure Governed Autonomous Edge Intelligence Architecture**, an open research initiative investigating architectural foundations for secure, governed, auditable, and trustworthy autonomous AI systems operating across distributed edge-cloud environments.

## Research Profiles

**Aridio Silva** — Independent Researcher, Brazil

- **ORCID:** https://orcid.org/0009-0008-2411-6995
- **Google Scholar:** https://scholar.google.com/citations?user=rPn5O48AAAAJ
- **Zenodo — SGAEIA Community:** https://zenodo.org/communities/sgaeia
- **OpenAIRE:** https://explore.openaire.eu/search/find?fv0=Aridio%20Silva&f0=q
- **Medium:** https://medium.com/@aridiosilva
- **GitHub:** https://github.com/aridiosilva
- **LinkedIn:** https://www.linkedin.com/in/aridio-silva-74997111/
- **SGAEIA Repository:** https://github.com/aridiosilva/SGAEIA
- **SGAEIA Research Artifact / DOI:** https://doi.org/10.5281/zenodo.22557796
- **Homepage:** https://aridiosilva.com
- **SGAEIA Homepage:** https://aridiosilva.com/sgaeia
- **SGAEIA LinkedIn:** https://www.linkedin.com/company/sgaeia/
- **Kaggle:** https://www.kaggle.com/aridiosilva2
- **DEV Community:** https://dev.to/aridiosilva
- **Academia.edu:** https://independent.academia.edu/AridioSilva
- **Article 23 — Zenodo / DOI:** https://doi.org/10.5281/zenodo.23732760
- **Article 23 — Academia.edu:** How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality
- **Article 23 — DEV Community:** How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality
- **Article 23 — Medium:** How to Reduce the Cost of AI Systems with LLMs Without Sacrificing Quality

## Figures

The cover is unnumbered. Figures 1–13 are numbered sequentially, cited in the text, and supplied in PNG and SVG. The conceptual assets remain within the public-disclosure boundary and are licensed under CC BY 4.0.

## License

Except where otherwise noted, the text and original conceptual illustrations in this article are licensed under the **Creative Commons Attribution 4.0 International License (CC BY 4.0)**.

© Aridio Silva @aridiosilva · October 2026. The SGAEIA software and research artifact remain subject to their own Apache License 2.0.

**Autonomous AI. Governed by Design. Trusted by Evidence.**

## Series continuity

This work is **SGAEIA Research Series — Article 23**. The public series catalogue is available at https://aridiosilva.com, with the project presence at https://aridiosilva.com/sgaeia. Persistent article record: https://doi.org/10.5281/zenodo.23732760.