

TECHNICAL REPORT · INFOGRAPHIC

Cutting LLM cost by 70% without losing quality

An evidence-based review of a six-lever reference architecture

Aridio Silva · Independent Researcher · ORCID 0009-0008-2411-5995

Analysis of R. Bhardwaj's proposal (LinkedIn, 26 Oct 2026) · 55 audited sources · 13 figures



Report roadmap

1 Introduction

2 Method

3 The central thesis

4 Cost model

5 The six levers

6 Reference architecture

7 Context engine

8 Caching: prompt, semantic and KV

9 Routing and cascades

10 Quality gates

11 Decomposition and sensitivity

12 Evaluation and KPIs

13 Observability and governance

14 Implementation roadmap

15 Threats to validity

16 Conclusion

The proposal holds; the 70% needs caveats

-70.7%

gross saving in the post: the five factors multiply

61-68%

plausible net saving once the levers' own overhead is added

±2.4 pts

sampling noise on a 500-item eval: larger than the reported gains

- All six levers have primary-source support, and several studies report larger effects than the post assumes.
- The 70% figure is illustrative: it leaves out routers, judges, embeddings and evaluation, and gives no confidence interval for quality.
- Recommended north-star metric: cost per successful task (cost-of-pass).

The object: an infographic in six parts

“

Cut LLM costs by up to 70%, not by compromising intelligence, but by engineering smarter model routing, context, caching and agentic workflows.

R. Bhardwaj, LinkedIn, 26 Oct 2026 (post text)

1

Headline

US\$10,000 → US\$2,926 per month

2

Six levers

routing, context, prompt caching, semantic cache, agent loop, batch

3

Architecture

six stages + a governance layer

4

Cost waterfall

five sequential cuts, labelled illustrative

5

Quality table

metrics before and after

6

Takeaway

spend intelligence only when the task needs it

Four steps, with every source audited

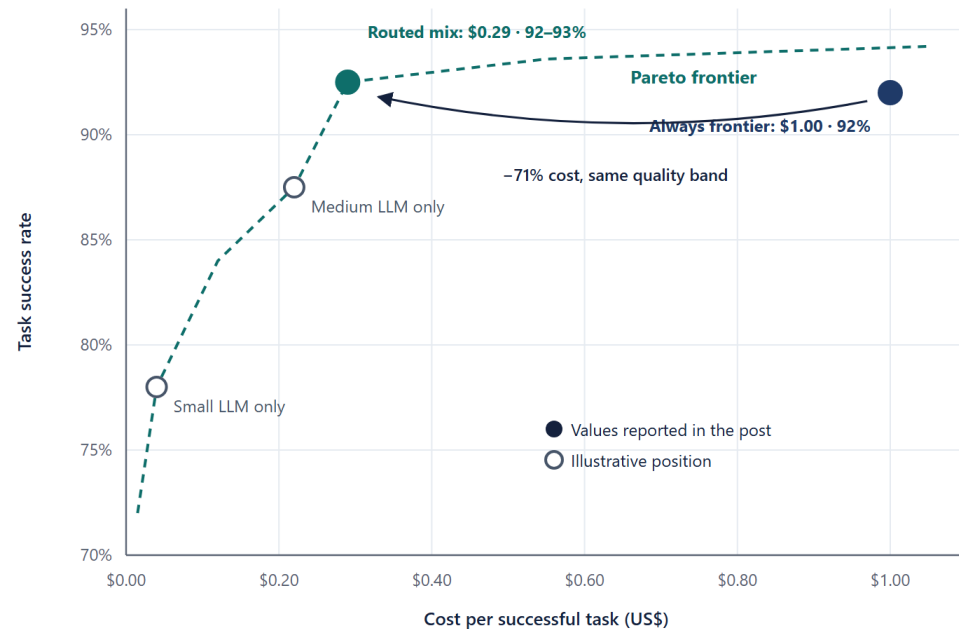


Limitations: a structured review, not a systematic one; no experiment was run; provider prices are quoted as read on the audit date.

Buy quality where it is needed

FIGURE 1

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



Cost-quality plane (conceptual)

Pareto frontier

Configurations where nothing is cheaper without also being worse.

Dominated point

Sending every request to the frontier model pays for capability most requests do not use.

Operating point

Fix a quality floor (SLO) first, then minimise cost per successful task.

Key takeaway

Spend more intelligence only when the task actually needs it.

How to read it

- Most real requests are easy: paying frontier prices for them buys idle capability.
- “Always frontier” is a dominated point: the routed mix reaches the same success band at 29% of the cost per success.
- Fix a quality floor (SLO) first, then minimise cost.
- Basis: Chen 2023; Kapoor 2024; Liang 2022.

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.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Cost per successful task

$$\text{CPST} = c / p$$

c = cost per attempt · p = success probability. Formalised as cost-of-pass (Erol et al. 2025).

Example

Model A: \$1.00 per attempt, 92% success → \$1.09 per success

Model B: \$0.10 per attempt, 46% success → \$0.22 per success

10× cheaper per attempt, only 5× per success, before handling its failures.

Strengths

- Matches router and cascade results.
- One business-readable metric.
- Works for agents as well as chat.

Limits

- Depends on the workload: if everything is hard, there is little to route.
- “Same quality” needs statistical proof.
- The critical long tail can degrade while the mean looks fine.

Each lever attacks one factor of the bill

FIGURE 2

Cost model: each lever attacks a different factor

Monthly LLM spend decomposed

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

N

Requests per month

Demand is given. The levers act on the cost of each request.

Not a lever

1 - h

Semantic-cache miss rate

A hit skips retrieval, generation and checks after one lookup.

L4 Semantic cache

k

LLM calls per request

Agent steps, tool loops, retries and duplicate calls.

L5 Agent loop

m → p(m)

Model chosen → token prices

API prices differed by up to two orders of magnitude (Chen et al. 2023).

L1 Routing

T_u

Uncached input tokens

Retrieved chunks, history and instructions billed at full price.

L2 Context

δ · T_c

Cached prefix tokens

Anthropic: reads δ = 0.1×, 5-min writes 1.25×.
OpenAI: up to 95% off.

L3 Prompt cache

T_{out}

Output tokens

Often priced several times the input rate. Cap with formats and max_tokens.

L2 Context

1 - β · b

Batch discount on batched share

β = 0.5 at Anthropic, OpenAI and Google. b = share that can wait up to 24 h.

L6 Batch

C_{ovh}

Overhead the levers add

Embeddings, vector DB, router, judges and evaluation runs.

New cost

Cost model by the author. Price multipliers: Anthropic, OpenAI and Google API documentation (accessed 2026-10-27).

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

How to read it

- N requests × (1 - h) semantic cache × k calls × model price × tokens × batch discount, + overhead.
- Each coloured card is one lever (L1–L6).
- The red card is the new cost the levers add, missing from the post.
- Cache and batch prices: Anthropic, OpenAI and Google docs (Oct 2026).

Savings multiply, they do not add up



Compounding

15% + 40% + 25% + 15% + 10% would add to 105%. Each cut acts on what remains, so the total is 70.7%.

Overlap

Once routing makes the model cheaper, cutting tokens saves at the lower price. Order changes each step's share, not the product.

Batch implies $b = 20\%$

With a $\beta = 50\%$ discount, the 0.90 factor = $1 - 0.5 \cdot b \Rightarrow$ one request in five can wait up to 24 h.

Target, evidence, risk and trade-off

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	$\delta \cdot T_c$	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	$\beta \cdot b$	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).

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Quick read

- Largest effect: routing (up to 98% in FrugalGPT).
- Lowest risk: prompt caching and batch.
- Highest quality risk: semantic cache (false hits).
- Next slides: one lever at a time.

Intelligent model routing

L1

Send each request to the cheapest model expected to answer it well. A cascade is the sequential variant: try cheap, check, escalate only on failure.

up to 98%

cost cut at GPT-4 accuracy

FrugalGPT · Chen 2023

>2×

cheaper with no quality loss

RouteLLM · Ong 2024

−40%

large-model calls, same quality

Hybrid LLM · Ding 2024

~40%

of GPT-4's cost on reasoning

MoT cascade · Yue 2023

Trade-off: routers need data and age as models change; a wrong “easy” verdict hurts the rare, high-stakes queries most.

Context engineering

L2

Decide which tokens enter the context window, and in what order: the smallest set of high-signal tokens for the task (Anthropic 2025).

middle

is where models use evidence worst

Lost in the middle · Liu 2024

up to 20×

prompt compression with little loss

LLMLingua · Jiang 2023

↑ tokens

degrade performance even on simple tasks

Context rot · Chroma 2025

Trade-off: cut too hard and you lose the passage that mattered (recall); compression adds a call and latency.

Prompt caching

L3

When many requests share a prefix (instructions, schemas, examples, a document), the provider reuses the processed state and charges less.

0.1×

cache-read price (5-min write 1.25×)

Anthropic docs 2026

up to 95%

off cached input; min. 1,024 tokens

OpenAI docs 2026

8×

faster time-to-first-token on GPU

Prompt Cache · Gim 2024

Trade-off: the prefix must be byte-identical; shared caches leak timing across users (7 providers audited, Gu 2025).

Semantic caching

L4

Store whole answers and return one when a new query is close in meaning to an earlier one, measured by embedding similarity.

~31%

of queries were repeats in the studied setting

MeanCache · Gill 2024

1 lookup

replaces retrieval, generation and checks on a hit

GPTCache · Bang 2023

3 risks

false hit, stale answer, cross-tenant leak

OWASP LLM Top 10 2025

Trade-off: it is the only lever that can return a wrong answer with no model involved, so it comes last in the roadmap.

Agent loop and batch processing

L5

Agent-loop optimisation (k)

- Every agent step is a paid call; redundant steps, retries and hand-offs multiply k.
- Simple baselines matched state-of-the-art agents at a fraction of the cost (Kapoor 2024).
- Start with the simplest pattern: one call, then a fixed workflow (Anthropic 2024).
- Trade-off: hard caps can stop an agent just before success; set them from traces.

L6

Batch processing ($\beta \cdot b$)

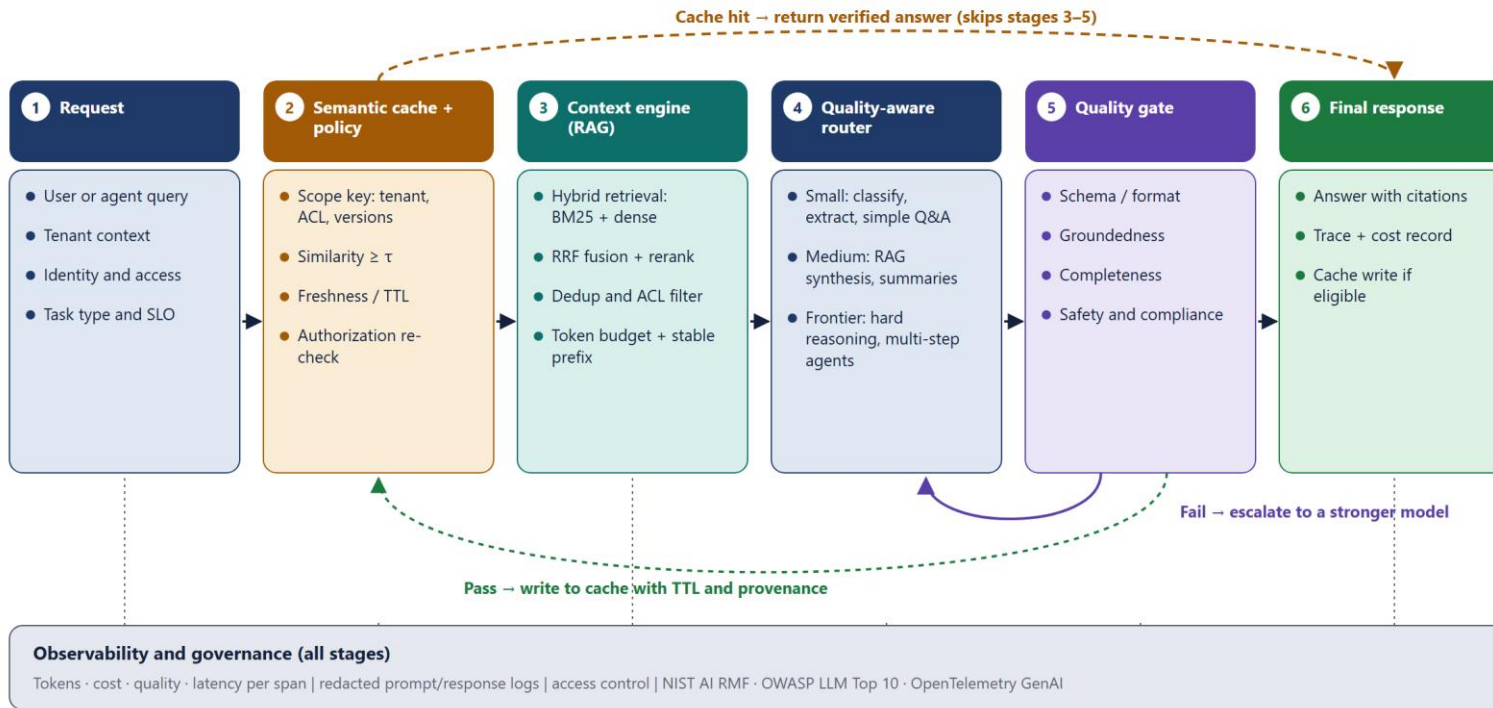
- Work that can wait goes in bulk, asynchronously: enrichment, backlogs, evaluations, reports.
- 50% discount and results within 24 h at Anthropic, OpenAI and Google.
- At Anthropic the discount stacks with prompt caching.
- Trade-off: only for latency-tolerant work; adds queueing.

Six stages and three control loops

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.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

The three loops

- **Amber:** a cache hit returns the verified answer and skips stages 3–5.
- **Violet:** if the gate fails, escalate to a stronger model.
- **Green:** only answers that passed the gate are cached, with TTL and provenance.
- **Basis:** Lewis 2020 (RAG); NIST AI RMF; OWASP 2025.

Three design principles



1

Cheap decisions first

Cache lookups and deterministic checks cost far less than a model call, so they run before it.

2

Close the loop

Only answers that passed the gate enter the cache, which keeps errors from being replayed.

3

Self-correcting retrieval

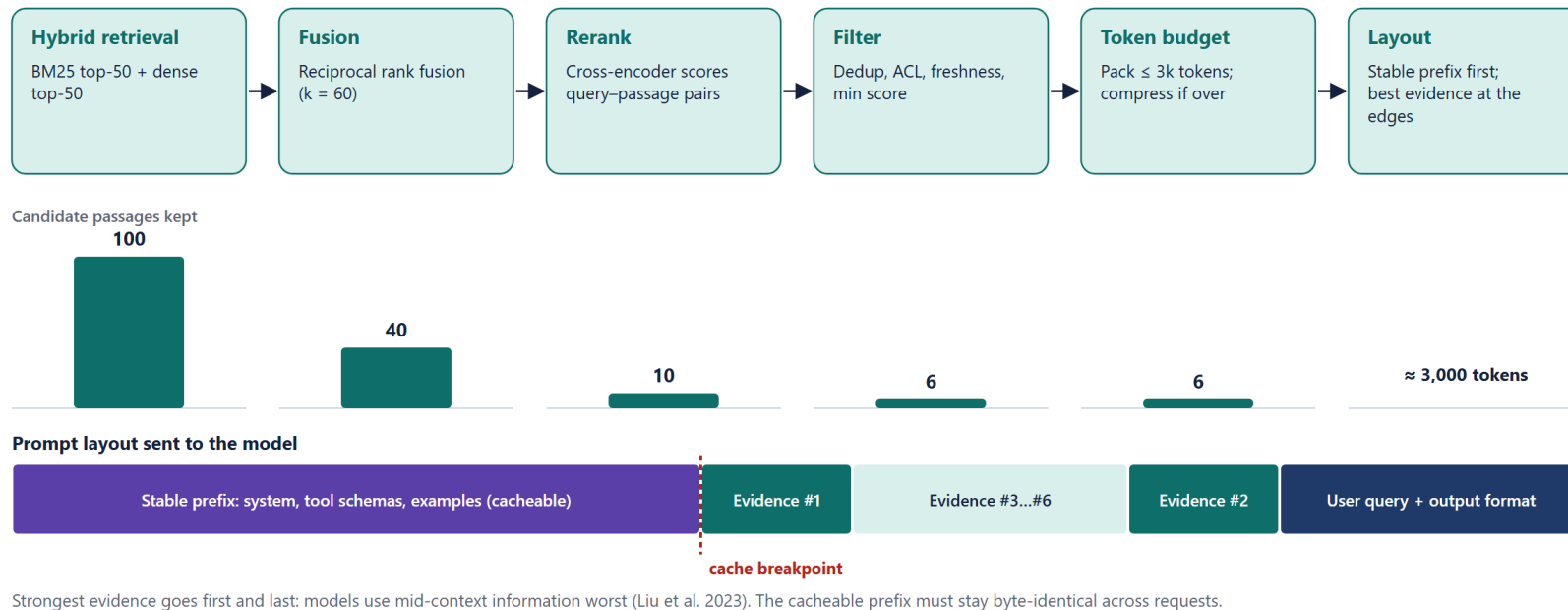
Self-RAG (Asai 2023) and Corrective RAG (Yan 2024) check retrieval inside the generation loop.

From 100 candidates to a budgeted prompt

FIGURE 5

Context optimisation engine: from 100 candidates to one budgeted prompt

Counts are illustrative



Sources: Robertson & Zaragoza 2009 (BM25); Cormack et al. 2009 (RRF); Nogueira & Cho 2019 (rerank); Liu et al. 2023; Jiang et al. 2023.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Steps

- BM25 finds exact terms; dense retrieval finds paraphrases.
- RRF merges the lists: $\Sigma 1/(60 + \text{rank})$, no score calibration.
- A cross-encoder reranks; filters drop duplicates, ACL and stale versions.
- Stable prefix first (cacheable); strongest evidence at the edges.

Three layers that are often confused

FIGURE 6
Three caching layers that are often confused

What each layer reuses and risks

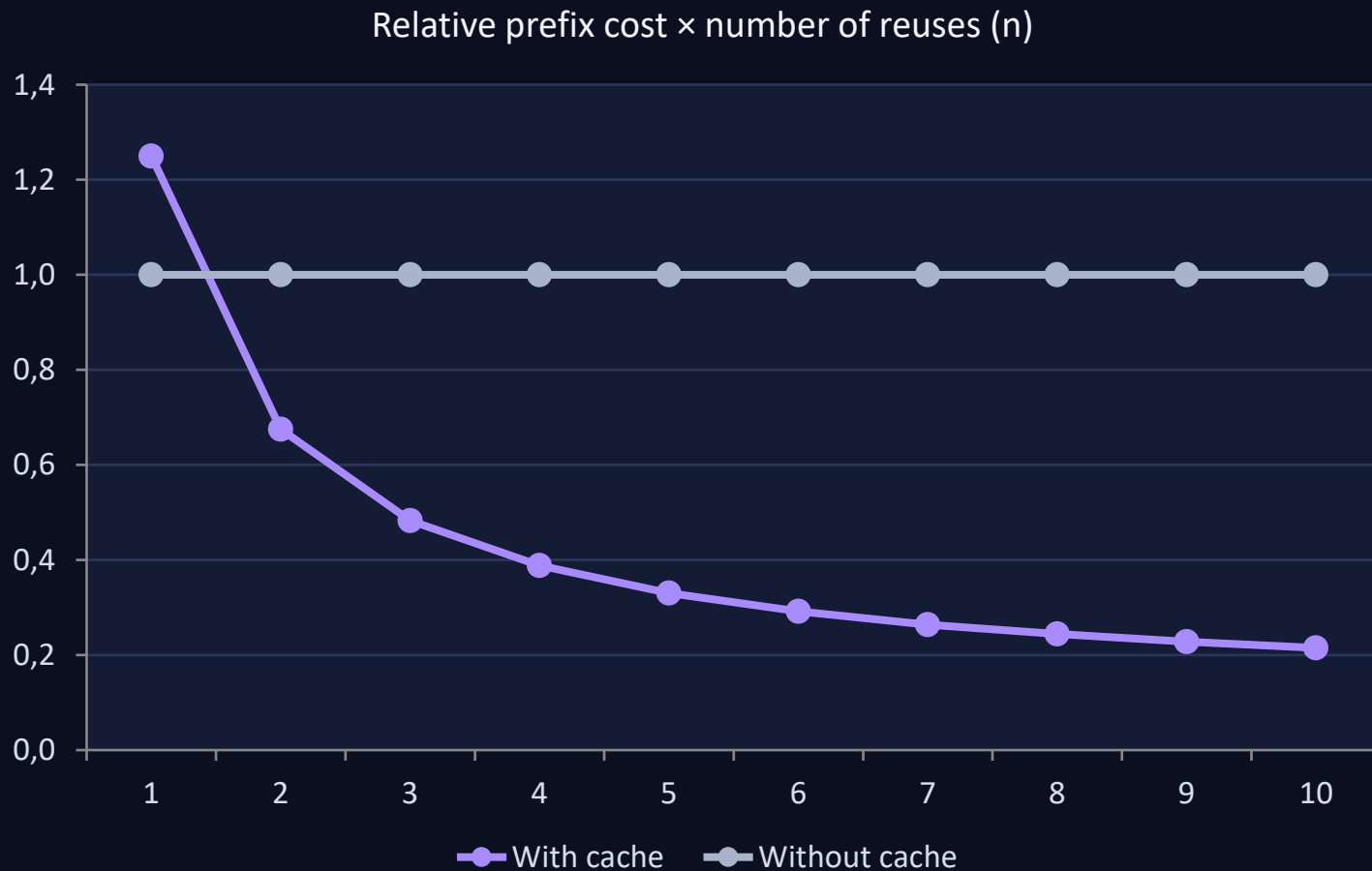
Layer	Reuses	Match rule	Saves	Main risk
Semantic cache Application layer (L4)	The whole final answer	Embedding similarity $\geq \tau$ inside a tenant/ACL/version scope	The entire pipeline: cost and latency	False hits, stale answers, cross-tenant leakage
Prompt / prefix cache Provider API (L3)	Processed prompt prefix (attention states)	Exact prefix match; $\geq 1,024$ tokens (OpenAI), explicit breakpoints (Anthropic)	Input price (reads $0.1\times$ 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).
© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

The difference

- Semantic: reuses the whole answer; biggest saving and biggest risk.
- Prompt cache: reuses the processed prefix; cheap and safe unless shared.
- KV cache: a concern for self-hosted models (PagedAttention, RadixAttention).

When prompt caching pays off



cost with cache / without

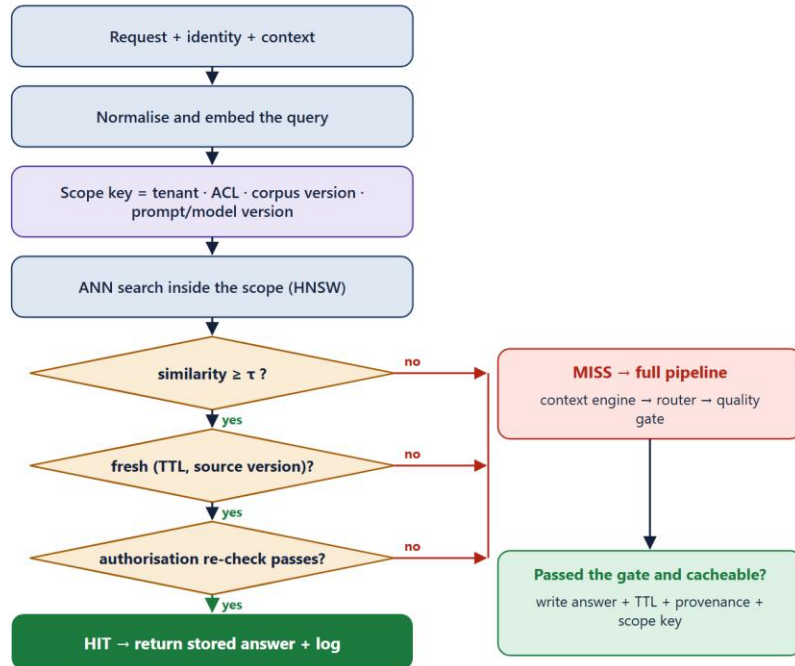
$$[1.25 + 0.1 \cdot (n - 1)] / n$$

- Pays from the 2nd use: 1.35 vs 2.0.
- At n = 10 the prefix costs 21.5% of full price.
- It loses money only if a written prefix is never reused before expiry.
- Anthropic multipliers: write 1.25×, read 0.1×.

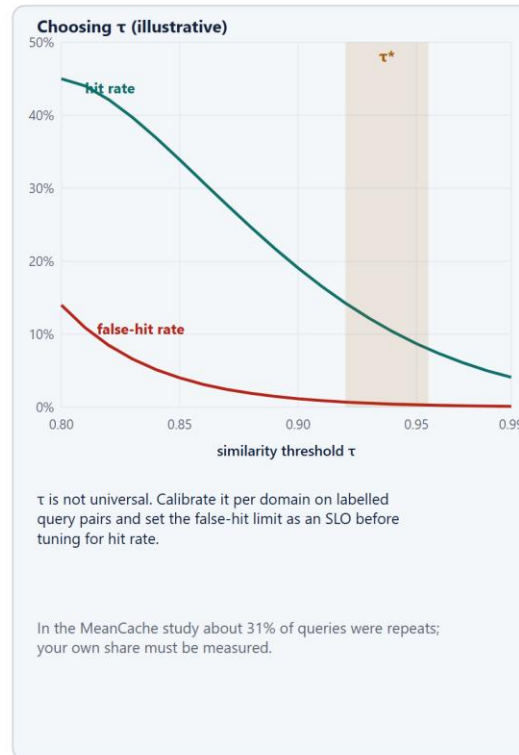
Similar, fresh and authorised

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.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Four controls

- Scope key: tenant, ACL, corpus and prompt version.
- Threshold τ calibrated per domain, with a false-hit limit as an SLO.
- Freshness: TTL plus invalidation when the source changes.
- Write only after the gate, with provenance.

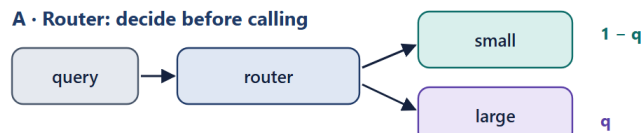
When a cascade stops paying

FIGURE 8

Routing vs. cascade, and when a cascade stops paying

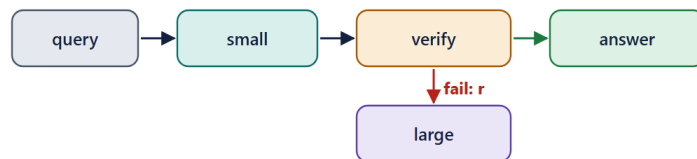
Expected cost relative to always using the large model

A · Router: decide before calling



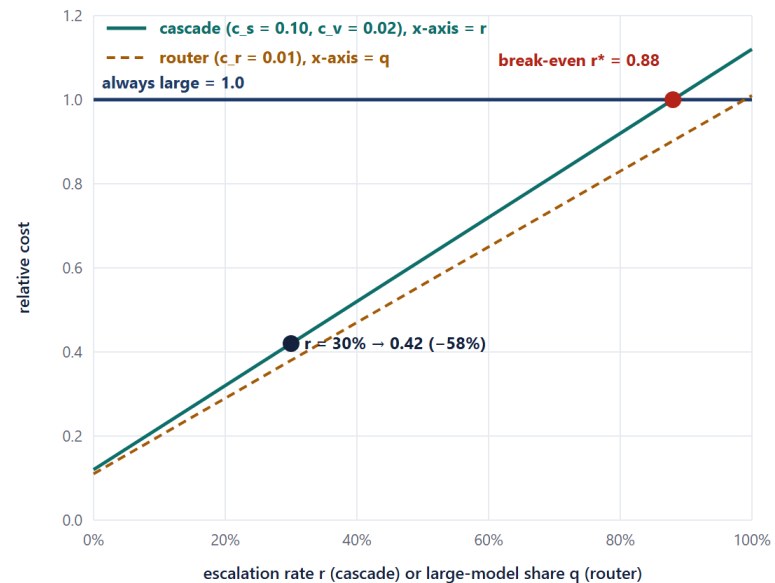
$$E[C_{\text{router}}] = c_r + (1 - q) \cdot c_s + q \cdot c_L$$

B · Cascade: try small, verify, escalate



$$E[C_{\text{cascade}}] = c_s + c_v + r \cdot c_L$$

Cascade saves money only while $r < 1 - (c_s + c_v)/c_L$. Escalated requests also pay the small model's latency.



The math

- Cascade: $E[C] = c_s + c_v + r \cdot c_L$.
- It pays only while $r < 1 - (c_s + c_v)/c_L$.
- With $c_s = 0.10$ and $c_v = 0.02$: break-even at $r^* = 0.88$.
- At $r = 30\%$: cost 0.42, or -58% .

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.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Router or cascade?

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

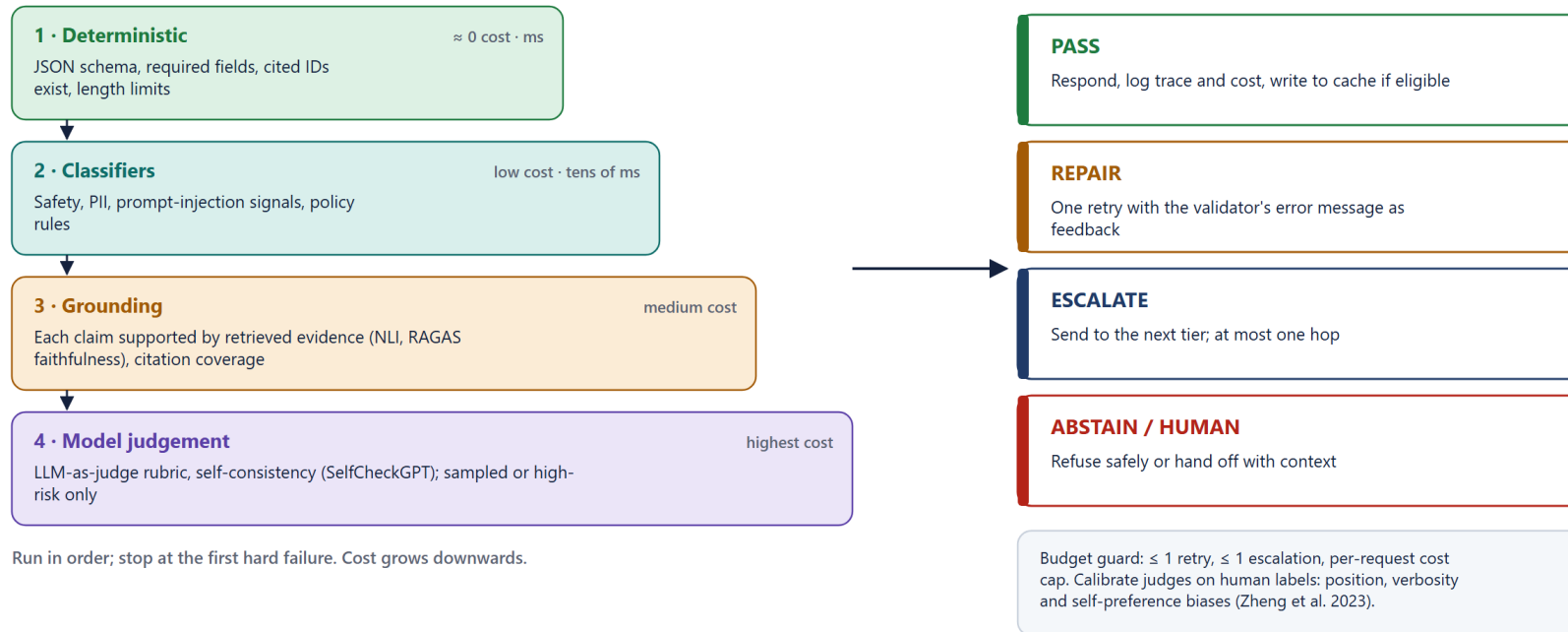
Signals used in the literature: predicted difficulty (Ding 2024), human preference (Ong 2024), answer consistency (Yue 2023), self-verification with a POMDP (Aggarwal 2023), token-level uncertainty (Gupta 2024).

Cheap checks first, bounded escalation

FIGURE 9

Quality gate: cheapest checks first, bounded escalation

Check tiers and outcomes



Sources: Es et al. 2023 (RAGAS); Manakul et al. 2023 (SelfCheckGPT); Zheng et al. 2023 (LLM-as-judge); Asai et al. 2023; Yan et al. 2024; OWASP 2025.

© 2026 Aridio Silva (@aridiosilva) • Independent Researcher • ORCID 0009-0008-2411-5995 • All figures licensed CC BY 4.0

Why it works

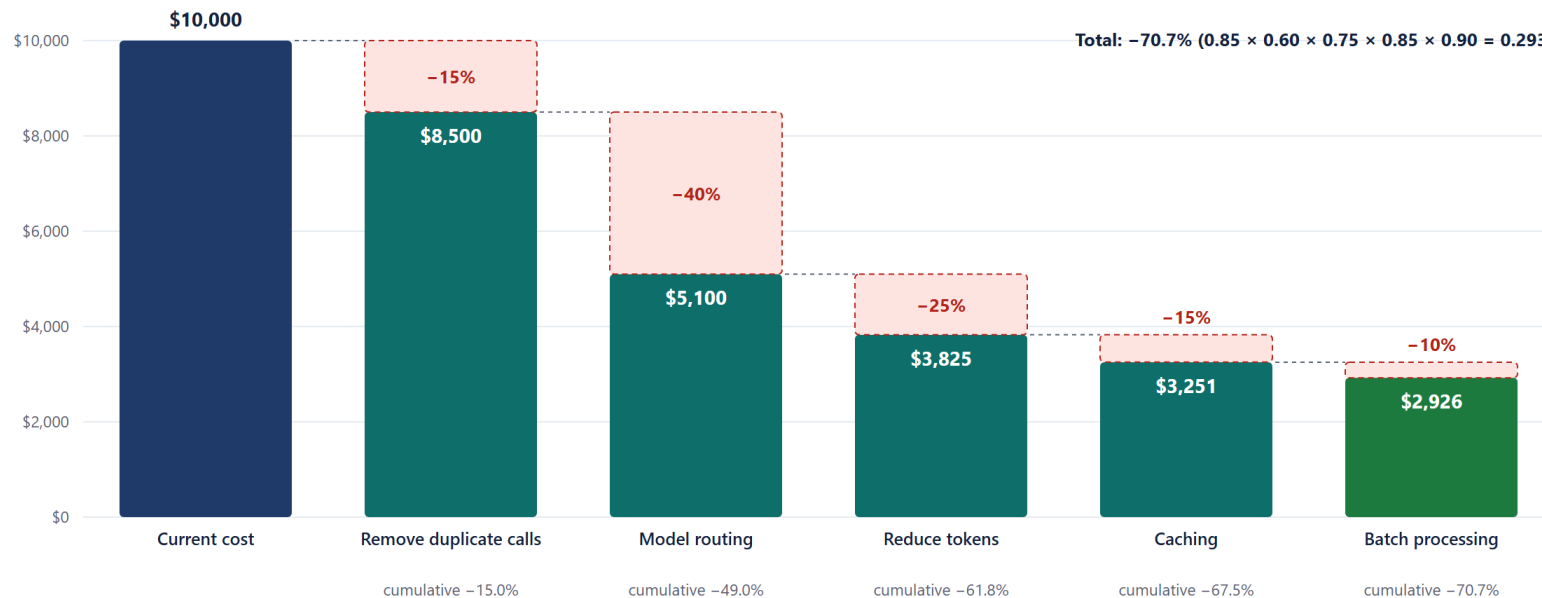
- The gate is what makes aggressive routing safe.
- Deterministic checks and classifiers catch most failures at almost no cost.
- LLM judges agree >80% with humans but show position, verbosity and self-preference biases (Zheng 2023).
- Guard: ≤ 1 retry, ≤ 1 escalation, a cost cap (OWASP LLM10).

The post's waterfall, re-drawn to scale

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.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Check

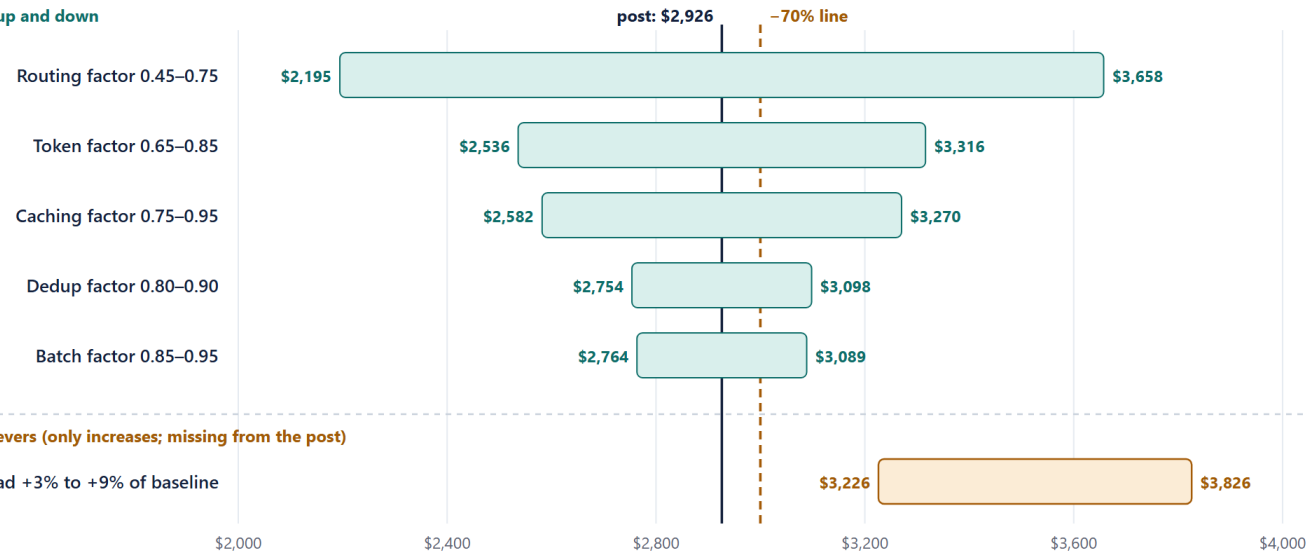
- $10,000 \times 0.85 \times 0.60 \times 0.75 \times 0.85 \times 0.90 = 2,926$: the arithmetic holds.
- Routing is the largest step: 34 of the 70.7 points.
- The post labels the chart illustrative: these are targets, not measurements.

How robust is the 70% claim?

FIGURE 11
Sensitivity: how robust is the 70% claim?

Final monthly cost, varying one assumption at a time

A • Post factors varied up and down



Routing alone moves the result from –64% to –78%. Adding realistic overhead puts the net saving at about 61–68%.

Model and ranges by the author. Rows A: one-at-a-time sensitivity around the post's step factors. Row B: overhead added on top. Illustrative.

© 2026 Aridio Silva (@aridiosilva) • Independent Researcher • ORCID 0009-0008-2411-5995 • All figures licensed CC BY 4.0

Result

- Routing alone moves the total from –64% to –78%.
- Batch and dedup matter least.
- The amber row (B) is not a variation: it is the cost the levers add, so it only goes up.
- With 3–9% of baseline as overhead, the net saving is ~61–68%.

What the levers add to the bill

Item	Introduced by	How to keep it small
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 judges
Cache writes (1.25×–2×)	L3	Cache only prefixes that will be reused
Evaluation runs	All	Run evals through the batch API (–50%)
Engineering and operations	All	Phase the work; quick wins first

With 3–9% of baseline in overhead, the net saving falls from 70.7% to about 61–68%.

“Same quality” needs statistical proof

±2.4 pts

95% interval for 92% success on 500 items

$$SE = \sqrt{(0.92 \cdot 0.08 / 500)} \approx 1.2 \text{ pts}$$

The post reports 92% → 92–93%, 90% → 90–91% and 0.5% → 0.4%: all inside the noise (Miller 2024).

1

Paired design

Evaluate both systems on the same items.

2

Non-inferiority margin

Pre-register, e.g. “success no more than 1 point lower”.

3

Sample size

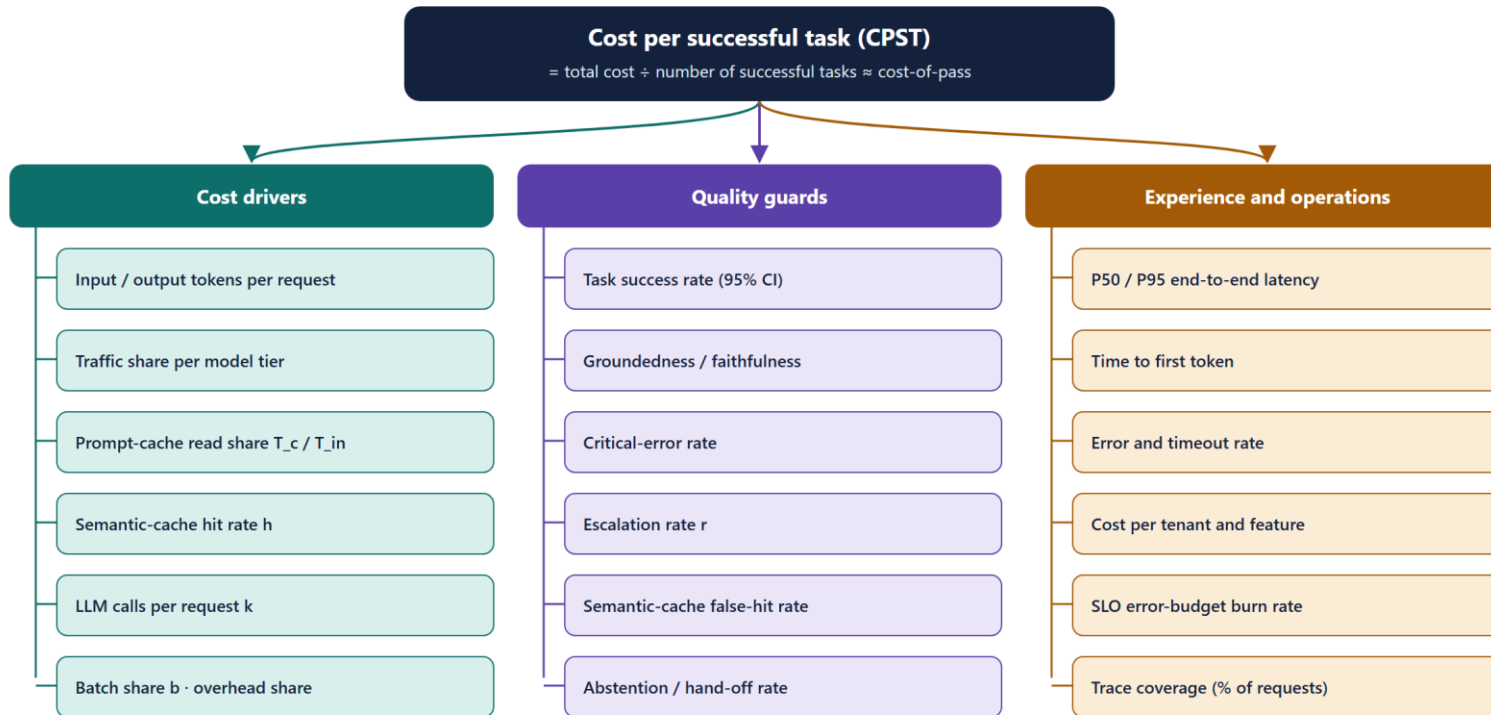
Size the set so the interval is narrower than the margin.

One north-star metric, three families

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.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

What to measure

- North star: cost per successful task (Erol 2025).
- Cost: tokens/request, model mix, cache read share, hit rate, k , batch.
- Quality: success with 95% CI, groundedness, critical errors, escalation, false hits.
- Operations: P50/P95, TTFT, timeouts, cost per tenant, trace coverage.

Without per-request telemetry, nothing can be tuned

1

OpenTelemetry GenAI

Standard span attributes for model, tokens and operations: cost per stage, portable across vendors.

2

NIST AI RMF + AI 600-1

Govern, Map, Measure and Manage functions, plus the Generative AI Profile.

3

OWASP LLM Top 10 2025

Prompt injection, sensitive information disclosure, vector and embedding weaknesses, unbounded consumption.

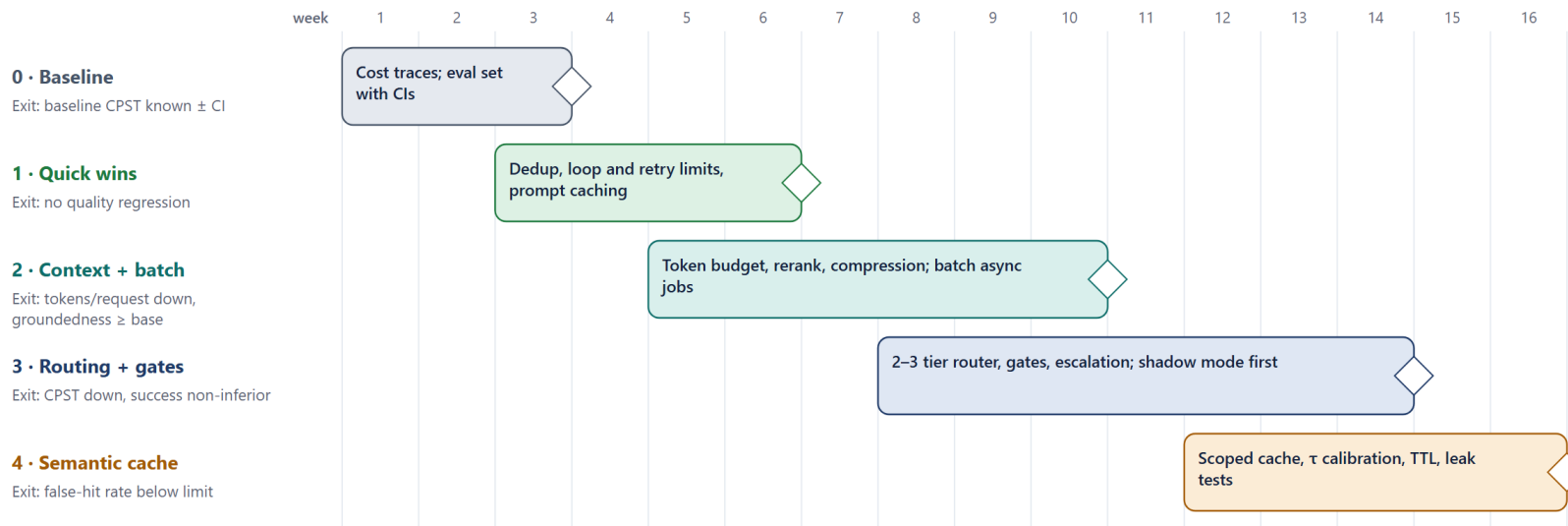
Hidden technical debt (Sculley 2015): every cache, router and judge is another component that can drift. Each needs an owner, a metric and an alert.

Low risk first, every phase gated

FIGURE 13

Implementation roadmap: low-risk levers first, each phase gated

Weeks are indicative



Roadmap by the author; sequencing follows Sculley et al. 2015, Kapoor et al. 2024, Miller 2024 and NIST AI 600-1.

© 2026 Aridio Silva (@aridiosilva) · Independent Researcher · ORCID 0009-0008-2411-5995 · All figures licensed CC BY 4.0

Order matters

- 0 · Measure: cost traces and an eval set with CIs.
- 1 · Quick wins: dedup, loop caps, prompt caching.
- 2 · Context and batch.
- 3 · Routing and gates, shadow mode first.
- 4 · Semantic cache last: it needs the most mature governance.

Where the numbers may not hold



Workload dependence

Hard, unique, multi-step tasks have little to route or cache.



Price volatility

Multipliers change several times a year; the size of the saving changes with them.



Latency

A lower P95 is plausible, but cascades add serial calls. Measure per route.



Evaluation validity

Routers tuned on the test set overstate gains.



Cache security

Shared caches create information channels between users.



Scope

A literature-based analysis; Figures 8 and 11 are illustrative models.

Spend intelligence only where the task needs it, and measure per success

Sound

All six levers have primary-source support, some with larger effects than the post assumes.

Realistic

A plausible net saving of 60–70% for workloads with many simple or repeated requests.

Safe

Measure first, take the low-risk levers, then route and cache behind gates and non-inferiority tests.

How to cite: Silva, Aridio. (2026). Cutting LLM cost by 70% without losing quality: an evidence-based review of a six-lever reference architecture (Version 1.0) [Technical report]. ORCID 0009-0008-2411-5995.



Key concepts

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.

Key concepts

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.

Key concepts

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.

Key concepts

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 $1/(k + \text{rank})$ 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 $\frac{3}{4}$ 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 on 27 Oct 2026

- [1] Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., et al. (2023). *AutoMix: Automatically mixing language models*. arXiv preprint arXiv:2310.12963. arxiv.org/abs/2310.12963
- [2] Anthropic (2024, December 19). *Building effective AI agents*. Anthropic Engineering. www.anthropic.com/research/building-effective-agents
- [3] Anthropic (2025, September 29). *Effective context engineering for AI agents*. Anthropic Engineering. www.anthropic.com/research/effective-context-engineering-for-ai-agents
- [4] Anthropic (2026). *Batch processing*. Claude Developer Platform documentation. platform.claude.com/docs/en/build-with-claude/batch-processing
- [5] Anthropic (2026). *Prompt caching*. Claude Developer Platform documentation. platform.claude.com/docs/en/build-with-claude/prompt-caching
- [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. arxiv.org/abs/2310.11511
- [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. aclanthology.org/2023.nlposs-1.24/
- [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. sre.google/sre-book/service-level-objectives/
- [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. www.linkedin.com/posts/rajeev-bhardwaj_genai-agentical-llmoptimization-share-7520411441674387456-qL-Q
- [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. arxiv.org/abs/2305.05176
- [11] Chroma Research (2025). *Context rot: How increasing input tokens impacts LLM performance*. Chroma technical report. www.trychroma.com/research/context-rot
- [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. doi.org/10.1145/1571941.1572114
- [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. arxiv.org/abs/2404.14618
- [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. arxiv.org/abs/2504.13359

References audited on 27 Oct 2026

- [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. arxiv.org/abs/2309.15217
- [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. arxiv.org/abs/2403.02694
- [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. arxiv.org/abs/2311.04934
- [18] Google (2026). *Batch API*. Gemini API documentation. ai.google.dev/gemini-api/docs/batch-api
- [19] Google (2026). *Context caching*. Gemini API documentation. ai.google.dev/gemini-api/docs/caching
- [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. arxiv.org/abs/2502.07776
- [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. arxiv.org/abs/2404.10136
- [22] Han, T., Wang, Z., Fang, C., Zhao, S., et al. (2024). *Token-budget-aware LLM reasoning*. arXiv preprint arXiv:2412.18547. arxiv.org/abs/2412.18547
- [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. arxiv.org/abs/2310.05736
- [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. arxiv.org/abs/2310.06839
- [25] Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., & Narayanan, A. (2024). *AI agents that matter*. arXiv preprint arXiv:2407.01502. arxiv.org/abs/2407.01502
- [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. arxiv.org/abs/2309.06180
- [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. arxiv.org/abs/2005.11401
- [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. arxiv.org/abs/2211.09110

References audited on 27 Oct 2026

- [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. arxiv.org/abs/2307.03172
- [30] LMSYS (2024–2026). *RouteLLM: A framework for serving and evaluating LLM routers [Software]*. GitHub. github.com/lm-sys/RouteLLM
- [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. arxiv.org/abs/1603.09320
- [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. arxiv.org/abs/2303.08896
- [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. arxiv.org/abs/2312.15234
- [34] Microsoft (2023–2026). *LLMLingua [Software]*. GitHub. github.com/microsoft/LLMLingua
- [35] Miller, E. (2024). *Adding error bars to evals: A statistical approach to language model evaluations*. arXiv preprint arXiv:2411.00640. arxiv.org/abs/2411.00640
- [36] National Institute of Standards and Technology (2023). *Artificial intelligence risk management framework (AI RMF 1.0)*. NIST AI 100-1. doi.org/10.6028/NIST.AI.100-1
- [37] National Institute of Standards and Technology (2024). *Artificial intelligence risk management framework: Generative artificial intelligence profile*. NIST AI 600-1. doi.org/10.6028/NIST.AI.600-1
- [38] Nogueira, R., & Cho, K. (2019). *Passage re-ranking with BERT*. arXiv preprint arXiv:1901.04085. arxiv.org/abs/1901.04085
- [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. arxiv.org/abs/2406.18665
- [40] OpenAI (2026). *Batch API*. OpenAI API documentation. developers.openai.com/api/docs/guides/batch
- [41] OpenAI (2026). *Prompt caching*. OpenAI API documentation. developers.openai.com/api/docs/guides/prompt-caching
- [42] OpenTelemetry (2026). *Semantic conventions for generative AI systems*. GitHub: open-telemetry/semantic-conventions-genai. github.com/open-telemetry/semantic-conventions-genai

References audited on 27 Oct 2026

- [43] OWASP GenAI Security Project (2025). *OWASP Top 10 for LLM applications 2025*. OWASP Foundation. genai.owasp.org/llm-top-10/
- [44] Ragas contributors (2023–2026). *Ragas: Evaluation toolkit for LLM applications [Software]*. GitHub. github.com/vibrantlabsai/ragas
- [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>. aclanthology.org/D19-1410/
- [46] Robertson, S., & Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond*. Foundations and Trends in Information Retrieval, 3(4), 333–389. doi.org/10.1561/15000000019
- [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. proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcdf2674f757a2463eba-Abstract.html
- [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. arxiv.org/abs/2304.09542
- [49] Yan, S.-Q., Gu, J.-C., Zhu, Y., & Ling, Z.-H. (2024). *Corrective retrieval augmented generation*. arXiv preprint arXiv:2401.15884. arxiv.org/abs/2401.15884
- [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. arxiv.org/abs/2210.03629
- [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. arxiv.org/abs/2310.03094
- [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. arxiv.org/abs/2306.05685
- [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. arxiv.org/abs/2312.07104
- [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. arxiv.org/abs/2404.14294
- [55] Zilliz (2023–2026). *GPTCache: Semantic cache for LLMs [Software]*. GitHub. github.com/zilliztech/GPTCache