Keeping the Cache Warm Pays: Keepalive Economics for Agentic Workloads

Maxim Khailo
blog.mempko.com
max@mempko.com

July 21, 2026

Abstract

Frontier LLM providers cache a prompt's processed prefix so that a follow-up request sharing it pays ∼10% of the input price and skips most of the prefill latency. Agentic workloads systematically destroy this benefit: the agent sends a request, runs a tool or waits for approval for minutes, and by the time the follow-up is sent the cached prefix has been evicted, so the agent pays the full prefill again. A client-side *keepalive*, replaying the prefix on a timer during the pause, prevents this, and it is *individually rational*: across Anthropic, OpenAI, Google, and DeepSeek we show that a keepalive holds the prefix warm through gaps where idle baselines are evicted, cutting the post-pause request cost by up to 12.5×. The strategic question is the ping *frequency*, and it has a clean answer: keepalive cost falls monotonically in the interval, so the economical choice is the largest interval safely under the provider's TTL, about 4 minutes at Anthropic's 5-minute TTL rather than the 30-second convention, and the strategy breaks even against a re-prefill at idle $\approx \tau(w/r - 1)$ (≈46 min for Anthropic, ≈36 min for OpenAI and DeepSeek). Because the benefit is real and bounded only by each user's own bill, rational adoption is universal adoption; and since cache residency is priced per read rather than per token-hour, a keepalive-saturated tier gives LRU eviction nothing to rank. We argue this externality will push providers to meter cache residency directly, and one already does. We derive the operator's policy until then.

1 Introduction

Modern LLM inference reuses the attention KV cache of a shared prompt *prefix*: a request beginning with recently processed bytes reads cached state instead of recomputing it [13, 8, 6]. Exposed as "prompt caching" [2, 10, 7], this makes cached input tokens roughly 10× cheaper and removes most of the prefill latency.

Agentic workloads are the worst-case user of this mechanism. An agent's loop is: think, act, wait. It sends a request, then runs a tool (a build, a test suite, a deployment, a wait for human approval) that takes minutes, and only then sends the follow-up that would have reused the (now large) conversation prefix. Provider caches expire in minutes: Anthropic's default TTL is five minutes [2], OpenAI clears entries after "5–10 minutes of inactivity" [10]. The pause outlives the cache, and the follow-up pays full price and full latency. At agent scale, dozens of such pauses per task and prefixes of 10^5 tokens, this is a real line item, and trace analyses of production coding agents show the same pattern from the serving side: long sessions repeatedly reuse large prefixes and put sustained pressure on the KV cache that generic serving policies handle poorly [14].

The defense is a *keepalive*: during the pause, re-send the exact prefix with minimal generation on a timer; each read refreshes the entry's TTL and its recency in the eviction policy. The technique is established practice [1, 3]; what has been missing is a quantified answer to the operator's actual questions. This paper provides it:

- **Does it work?** Yes, everywhere we measured. Across Anthropic, DeepSeek, Google, OpenAI, prefix sizes of 40k, 100k tokens, and idle gaps to 600 s, a 30 s keepalive held a median of ≥98% of prompt tokens cached in every provider-size-idle group; at 600 s, where Anthropic's baseline went 0/48 warm across three runs, the keepalive held 40/40 (§5.1).

- **What frequency is economical?** Not the 30 s convention. Keepalive cost falls monotonically in the interval, so the optimum is the largest interval safely under the TTL: ∼4 min at Anthropic, an 8× reduction in keepalive spend (§5.2).

- **How far does the benefit extend?** To idle $\approx \tau(w/r - 1)$, about 46 min at Anthropic's prices. Past that, warmth costs more than the re-prefill it prevents; below the TTL it buys nothing (§5.2).

- **What happens when everyone does it?** The incentive is individual and immediate, so adoption is rational; but cache residency is priced per read, not per token-hour, so a keepalive-saturated tier degrades for everyone. We argue providers will be forced to meter residency (Google's explicit cache already bills per token-hour), and we derive the builder's policy until then (§6).

2 Background and provider parameters

Anthropic caches on an explicit `cache_control` breakpoint: writes bill at 1.25× input, reads at 0.1×, default TTL 5 minutes with a paid 1-hour tier at 2× write [2]. OpenAI caches automatically above ∼1024 tokens at 0.1× cached-read price, clears after 5–10 minutes idle (longer off-peak), routes by prefix hash, and offers `prompt_cache_key` for routing affinity [10]. Google caches implicitly above ∼2048 tokens at ∼0.25× cached-read price [7]. DeepSeek caches automatically at 0.1× read price; its first-party cache is documented as disk-backed. In all four, *reading a cached prefix refreshes it.* Table 1 collects the parameters that drive the economics; the keepalive's whole mechanism is the refresh-on-read semantics in the last row.

The keepalive in production. We implement the defense in the `pi` agent harness as a per-tool-call keepalive: pings run only while a tool batch executes and stop when it completes (default off). The bounded form is the economically correct form (§5.2).

3 Related work

Keepalive-style cache management is shipping practice: Aider added a cache keepalive in 2024 [1], and Anthropic's documentation recommends periodic pre-warming [3]. Practitioners have also independently derived pieces of the economics we quantify: community cost analyses of Anthropic's 5-minute TTL [18], a documented 4-minute ping interval with a single-scenario break-even estimate [15], and keepalive plugins and proposals with break-even arithmetic [4, 11]. These are single-provider (Anthropic), single-scenario, and computed rather than measured. On the academic side, an evaluation of cache *placement* for agentic tasks [9] studies what to cache rather than how long it lives. On the systems side, prefix-cache eviction is documented in vLLM's design (LRU over content-addressed blocks) [16], studied for batched inference in BatchLLM [19], characterized at production scale in *KVCache Cache in the Wild* [17], and studied for coding agents specifically in CacheWise [14], whose trace analysis motivates reuse-aware eviction on the serving side. Our question is the client-side complement of theirs: given caches we cannot see, what can the agent itself do? CacheProbe studies cache isolation across OpenRouter accounts [5]; the multi-tenant billing externality of §6 was described informally as "prompt cache thrashing" [12]. What is missing is the measured, cross-provider account: retention and keepalive efficacy across four providers with timing-gated data, whole-strategy costs including the pings, and a per-provider policy. That is our contribution.

4 Method

Harness. For each provider we send, per measurement *cell*: reqA, a large prefix with a unique per-cell salt (writing the cache); an idle gap of I seconds, in the keepalive condition replaying the identical prefix every 30 s; and reqB, the identical prefix, streamed to capture time-to-first-token. Anthropic, OpenAI, and Google are queried first-party (Anthropic Messages API with an explicit ephemeral breakpoint; OpenAI Chat Completions with `prompt_cache_key` set to the cell salt; Gemini with implicit caching). DeepSeek is queried via OpenRouter pinned to a single backend (DeepInfra); the pin is endpoint-level, so the served backend is recorded on every call and any cell whose backend changes mid-cell is discarded.

Timing integrity. Every call records queue-entry, start, and completion timestamps, and the quantity analyzed is the *true idle* experienced by the cache entry (`reqB.start − reqA.end`), not the nominal schedule. Cells launch staggered in bounded shuffled batches so offered load stays well under the HTTP concurrency limit; a cell is valid only if reqB's local queue wait is ≤5 s and the true idle is within 10 s of nominal. Every run contains idle = 0 warm-reference cells, an immediate re-read of a just-written prefix, and a run whose warm references miss is flagged and its timing re-verified before inclusion. Baseline and keepalive cells run in *separate time blocks* (order alternated across replicates) so keepalive traffic cannot pressure the tier during baseline measurement. In the runs reported here, median queue wait was 0 ms and median idle slip <1 s across the valid cells.

Cost accounting. Every call (reqA, every keepalive ping, reqB) is priced and recorded. First-party responses carry no cost field, so cost is computed from usage and public list prices; DeepSeek uses OpenRouter's reported cost. Strategy cost is reported both per-request (reqB) and whole-cell (reqA + pings + reqB).

Matrix and statistics. We test Anthropic, DeepSeek, Google, OpenAI at 40k, 100k tokens, idle gaps of {0, 60, 300, 600} s, n=8 samples per cell per run, in three independent runs separated by 30 minutes to hours. The data is bimodal (a prefix is essentially warm or evicted), so we report the *warm rate*: the fraction of valid samples whose reqB cached ≥90% of prompt tokens. Samples within a run share one moment's tier conditions, so the run is the unit of independence: per-run Fisher exact tests (Mann–Whitney as a continuous check), Bonferroni-corrected, gated on a ≥10 percentage-point effect, and a claim is credited only if it holds in every run.

Table 1: Parameters driving keepalive economics (list prices, July 2026): cached-read ratio r and re-prefill ratio w relative to input price, documented idle TTL T, and the derived economical interval τ^* and break-even horizon $I_{\max}$ (§5.2). Google's higher r shrinks its break-even horizon.

Provider	r	w	T	τ^*	$I_{\max}$	rent/hour at τ^* (100k prefix)
Anthropic (5-min tier)	0.10	1.25	300 s	∼240 s	≈46 min	$0.45
Anthropic (1-hour tier)	0.10	2.00	3600 s	∼50 min	≈3.3 h	$0.04
OpenAI	0.10	1.00	300–600 s	∼240 s	≈36 min	$0.19
Google (implicit)	0.25	1.00	minutes	∼240 s	≈12 min	$0.47
DeepSeek	0.10	1.00	≤600 s	∼240 s	≈36 min	$0.04

Table 2: Fraction of samples whose post-idle `reqB` was warm (cache hit ≥ 90%), baseline vs. keepalive. Only cells passing all measurement-integrity gates are counted; conditions were collected in separate time blocks.

Provider	Size	Idle	Warm$_{base}$	Warm$_{ka}$
Anthropic	40k	60	24/24	24/24
Anthropic	40k	300	24/24	21/21
Anthropic	40k	600	0/24	20/20
Anthropic	100k	60	24/24	24/24
Anthropic	100k	300	24/24	19/19
Anthropic	100k	600	0/24	20/20
DeepSeek	40k	60	20/24	20/24
DeepSeek	40k	300	16/23	21/21
DeepSeek	40k	600	3/24	21/21
DeepSeek	100k	60	17/24	19/24
DeepSeek	100k	300	20/24	21/23
DeepSeek	100k	600	1/24	21/21
Google	40k	60	12/12	6/7
Google	40k	300	7/12	5/5
Google	40k	600	11/12	6/6
Google	100k	60	8/12	7/7
Google	100k	300	8/12	5/5
Google	100k	600	9/12	3/3
OpenAI	40k	60	24/24	24/24
OpenAI	40k	300	22/24	20/21
OpenAI	40k	600	20/24	23/23
OpenAI	100k	60	24/24	24/24
OpenAI	100k	300	23/24	24/24
OpenAI	100k	600	19/24	23/23

5 Results

5.1 The keepalive works

Figure 1 and Table 2 are the efficacy result, and the four providers sort into three regimes. *Hard TTL* (Anthropic): the baseline is warm through 300 s and evicted to 0/48 samples across the three runs at 600 s, while the keepalive holds 40/40 at the same gap; the separation is Bonferroni-significant in every run at both prefix sizes. *Soft, lossy cache* (DeepSeek via DeepInfra): the baseline evicts (4/48 warm at 600 s) and the keepalive holds 42/42, but short-idle samples show scatter we attribute to machine-level routing inside the pinned endpoint, including occasional keepalive misses at 60 s that first-party providers did not produce; an endpoint pin is not a machine pin, and a ping can warm one machine while reqB lands on another. *Sticky cache* (OpenAI, Google): baselines survive 600 s in most samples (OpenAI 39/48; Google 20/24 across its two completed runs), so at these gaps there is little to defend and the keepalive's role is variance removal. Google's implicit cache also carries the one diurnal signal in our data: its 600 s baseline was 12/16 in the peak-evening run and 8/8 in the off-peak night run, matching the "longer off-peak" behavior OpenAI documents for its own cache. At 60–300 s, Anthropic and OpenAI baselines are fully or nearly fully warm, while DeepInfra and Google already lose a third or more of samples at 300 s in the peak run. Our strict bar (Fisher exact per run, Bonferroni over the 24 comparisons, required in every run) is met by Anthropic at both sizes and DeepSeek at 100k; DeepSeek at 40k shows the same pooled separation but falls one run short of the per-run bar. Across all providers, sizes, and idles the keepalive's median cache hit is ≥98%, and the ping logs show the schedule was actually kept (median ping drift <0.2 s).

5.2 Keepalive economics: frequency and horizon

Let r be the cached-read price ratio, w the re-prefill ratio (1.25 at Anthropic, 1.0 where re-caching is automatic and free), T the TTL, and τ the ping interval. Keeping a prefix alive through an idle I costs $(I/\tau + 1)\, r$ per input token; letting it die costs w once. Everything operators ask follows from this.

Frequency: ping as rarely as the TTL allows. Keepalive spend per unit time is r/τ, strictly decreasing in τ, and the interval purchases nothing except TTL safety. The economical interval is therefore $\tau^* = T - \text{margin}$, where the margin covers ping latency, jitter, and TTL-enforcement slack: ∼4 min against Anthropic's 5-minute TTL (Table 1), an interval previously suggested in practitioner work [15, 11] and which we validate experimentally below. The 30 s convention in circulation (including our own efficacy runs above) spends 8× more than necessary (7.8× measured across our interval runs): holding a 100k-token Anthropic prefix costs ∼$3.60/hour at 30 s pings versus ∼$0.45/hour at τ^*. Where the TTL is fuzzy (OpenAI's "5–10 minutes"), the uncertainty belongs in the margin, not the interval.

Horizon: the benefit has a ceiling. Break-even against re-prefill is $I_{\max} = \tau\,(w/r - 1)$: ≈46 min for An-

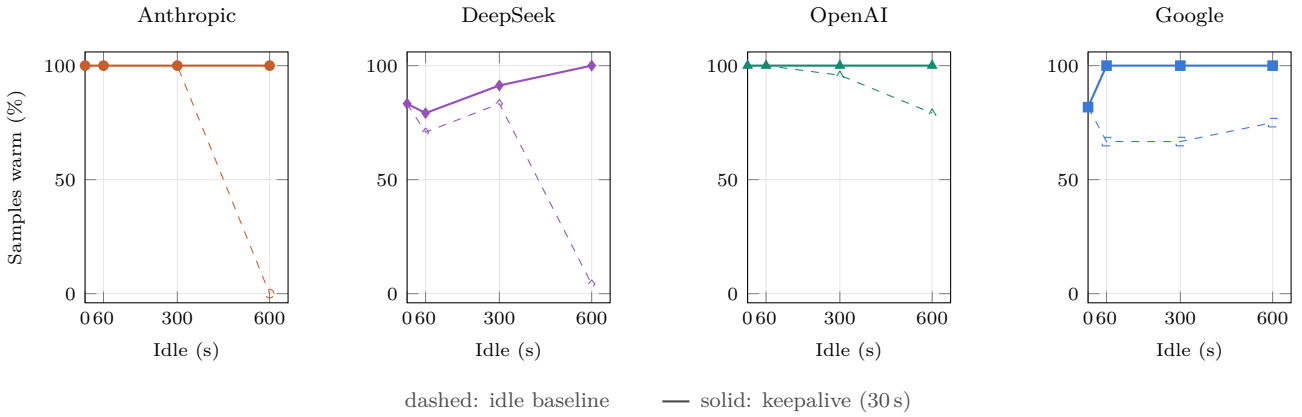

Figure 1: **Per-provider warm rate vs. idle** (fraction of valid samples whose post-idle reqB cached ≥90% of prompt tokens; 100k prefix; 40k shows the same pattern on Anthropic, DeepSeek, and OpenAI). The three regimes read off directly: Anthropic's baseline collapses at 600 s (hard TTL) while the keepalive holds; DeepSeek's baseline is lossy at every gap; OpenAI's and Google's baselines barely need defending. Google's keepalive points are n=3–6 (quota-limited); all other points are up to n=24 across three runs.

thropic's 5-minute tier at τ^*, ≈36 min for OpenAI and DeepSeek, and only ≈12 min for Google, whose implicit cache reads at 0.25× rather than 0.1× (Table 1). Below the TTL the keepalive buys nothing (the baseline survives anyway); past $I_{\max}$ it costs more than the eviction it prevents. Table 3 shows this directly at our 600 s gap: on the post-pause request alone the keepalive is up to 12.5× cheaper (anthropic at 100000 tokens, 600s idle (post-idle request only)), but *net of its own pings* the 30 s keepalive costs more than the cold re-prefill on every provider and size where both arms were measured, because $I_{\max}$ at 30 s is only ≈6 min. The τ^*≈240 s arm cuts ping spend 7.8× against the 30 s arm (measured) while holding 23 of 24 samples warm (the one miss on the lossy DeepInfra endpoint), and the net savings materialize exactly where the formula says they should: 1.6× on Anthropic at both sizes (hard TTL, 1.25× write premium), but not on DeepSeek (0.7–0.8×; its re-prefill is too cheap to insure), OpenAI (0.84×; nothing to insure against in this window), or Google (0.6–0.7×; a sticky baseline *and* 0.25× reads, the worst case for the strategy). Paid long-TTL tiers raise the ceiling into the hours: Anthropic's 1-hour tier breaks even against a 1.25× re-prefill at ∼3.3 h net of its write premium. No configuration makes indefinite warmth economical: the rent always exceeds the re-prefill eventually.

Latency: the unqualified win. Cost is the bounded benefit; latency is not. At 600 s, warm reqB TTFT beats the cold re-prefill on every provider where eviction occurs (Table 3): ∼0.8 s faster on a 100k Anthropic prefix and ∼4 s on 100k DeepSeek. For interactive agents, this alone can justify the keepalive even past $I_{\max}$.

5.3 The operator's policy

Keep the cache warm across pauses whose length is bounded and plausibly reused (a tool call, an approval wait), at τ^*; abandon the keepalive when the pause exceeds $I_{\max}$; use the paid long-TTL tier for hour-scale gaps; never keep a dead session warm. This is the policy `pi` implements.

6 Discussion: rational adoption, and the provider response it forces

We expect readers to implement keepalives, and they would be rational to: the benefit is immediate, the cost is bounded by their own bill, and no provider terms forbid it; providers in fact recommend pre-warming [3]. But the equilibrium this points at is degraded, and the mechanism is worth stating precisely. A cache tier's eviction policy ranks by expected reuse; a keepalive manufactures recency, so once every client keeps its prefixes alive, LRU has nothing left to rank and the tier degrades toward first-in-first-out-of-luck. Residency today is priced per *read*, not per token held per second, and the price does not rise when the tier is hot. Each operator's rational r/τ rent payment therefore imposes an unpriced congestion cost on every other tenant: shorter effective TTLs, lower hit rates, and the "prompt cache thrashing" spiral in which keepalive becomes mandatory to extract any cache value at all [12]. Individually rational, collectively self-defeating.

Why do providers encourage the practice today? Because at current adoption the exchange is paid and bounded: each refresh bills at the read rate (holding a 100k prefix alive at τ^* costs the client ∼1.5× the input price per hour, recurring rent for the memory), the

Table 3: Measured whole-strategy cost at the longest idle gap (reqA + pings + reqB, median, with warm rate): letting the cache die and re-prefilling (baseline) versus the 30 s keepalive convention and the economical $\tau^* \approx 240\,s$ prescription. The 30 s keepalive loses money at this gap on every provider; the 240 s keepalive keeps the warmth benefit at a fraction of the ping spend. Valid cells only.

		baseline			keepalive 30 s			keepalive 240 s		
Provider	Size	cost	warm	TTFT	cost	warm	TTFT	cost	warm	TTFT
Anthropic	40k	\$0.267	0/24	1654 ms	\$0.347	24/24	1240 ms	\$0.166	4/4	1267 ms
Anthropic	100k	\$0.667	0/24	2285 ms	\$0.867	24/24	1453 ms	\$0.414	4/4	1332 ms
DeepSeek	40k	\$0.017	3/24	2332 ms	\$0.100	25/25	1039 ms	\$0.022	4/4	1388 ms
DeepSeek	100k	\$0.043	1/24	5398 ms	\$0.249	25/25	1382 ms	\$0.060	3/4	1950 ms
Google	40k	\$0.053	11/12	1846 ms	\$0.291	10/10	1429 ms	\$0.090	3/4	2117 ms
Google	100k	\$0.131	9/12	3952 ms	\$0.649	7/7	1800 ms	\$0.186	3/4	2561 ms
OpenAI	40k	\$0.046	20/24	1007 ms	\$0.130	27/27	901 ms	\$0.055	4/4	1074 ms
OpenAI	100k	\$0.115	19/24	1135 ms	\$0.317	27/27	1051 ms	\$0.136	4/4	1508 ms

provider retains the right to evict, and a cache-read ping converts a compute-expensive re-prefill into a nearly free memory read, smoothing their compute demand. But the levers for the congested regime are already visible, and we predict they will be pulled as keepalive adoption spreads: absolute lifetime caps, per-account residency quotas, paid long-TTL tiers (Anthropic's 1-hour tier at 2× write is an early step), and, the clean solution, metering residency per token-hour, which Google's explicit context cache already does [7]. Token-hour pricing kills speculative warmth while leaving real reuse profitable: the rent then tracks the resource actually consumed. The economic opportunity this paper quantifies is therefore best understood as an arbitrage with an expiry date: profitable now, and self-extinguishing at scale. Builders who adopt the bounded policy of §5.2 keep the benefit under every regime we can foresee, including the metered one.

7 Threats to validity

- **Backend identity.** DeepSeek numbers describe the pinned OpenRouter backend (DeepInfra), whose cache behavior is its own; an endpoint pin is not a machine pin, which we observe as occasional keepalive misses at short idles.
- **Within-run correlation.** Samples in a run share one tier moment; per-run p-values are descriptive, and only across-run replication is claimed.
- **Google's quota.** Gemini cells were collected under a restrictive account request quota (1K requests/day, which keepalive blocks repeatedly exhausted): Google has two complete matrix runs (peak evening and off-peak night) and one interval-validation run, plus partial cells from two quota-killed runs; its keepalive arms remain thin (n=3–7 per cell). Its implicit cache also offers no affinity lever, so warm rates carry machine-lottery variance on top of the ordinary kind.
- **Price drift.** Dollar figures use July-2026 list prices; ratios and the form of the results are robust to drift.
- **Load.** Retention timescales are measured at the load one measurement client generates; provider tiers under fleet-wide keepalive pressure are the subject of §6, not of our retention curves.

8 Conclusion

In agentic workloads the pause outlives the cache, and the keepalive, reading the prefix back on a timer, is a proven, individually rational defense: it holds prefixes warm across Anthropic, OpenAI, Google, and DeepSeek through gaps that evict idle baselines, and it cuts the post-pause request cost by up to 12.5×. The economical ping frequency is the largest interval safely under the TTL, about 4 min at Anthropic rather than the 30 s convention, and the saving has a hard ceiling at $\approx\tau(w/r-1)$, tens of minutes at current prices. Because the benefit is real, universal adoption is the rational equilibrium; and because residency is not metered, that equilibrium degrades the shared tier until providers price cache residency directly. One already does. The rest, we predict, will follow, and the sooner operators understand the incentive, the sooner that day arrives.

References

[1] Aider AI. Aider change history (v0.53.0: cache keepalive). `https://github.com/Aider-AI/aider/blob/main/HISTORY.md`, 2024. Accessed 2026.

[2] Anthropic. Prompt caching. `https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching`, 2024. Accessed 2026.

[3] Anthropic. Prompt caching: keeping the cache warm. `https://platform.claude.com/docs/en/build-with-claude/prompt-caching`, 2026. Accessed 2026.

[4] Yujia Chen. claude-code-cache-keepalive (plugin, with break-even analysis). `https://github.com/yujiachen-y/claude-code-cache-keepalive`, 2026. Accessed 2026.

[5] Ryan Fahey. Cacheprobe: Auditing prompt cache isolation in gateway apis. *arXiv preprint arXiv:2605.30613*, 2026.

[6] In Gim, Guojun Chen, Seung-seob Lee, et al. Prompt cache: Modular attention reuse for low-latency inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024.

[7] Google. Context caching. `https://ai.google.dev/gemini-api/docs/caching`, 2024. Accessed 2026.

[8] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, et al. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM Symposium on Operating Systems Principles (SOSP)*, 2023.

[9] Elias Lumer, Faheem Nizar, Akshaya Jangiti, Kevin Frank, Anmol Gulati, et al. Don't break the cache: An evaluation of prompt caching for long-horizon agentic tasks. *arXiv preprint arXiv:2601.06007*, 2026.

[10] OpenAI. Prompt caching. `https://platform.openai.com/docs/guides/prompt-caching`, 2024. Accessed 2026.

[11] OpenClaw. Feature: prompt cache keep-warm pings (issue #62475). `https://github.com/openclaw/openclaw/issues/62475`, 2026. Accessed 2026.

[12] Tian Pan. Prompt cache thrashing: a multi-tenant noisy-neighbor billing scenario. `https://tianpan.co/blog/2026-04-28-prompt-cache-thrashing-multi-tenant-noisy-neighbor`, April 2026. Accessed 2026.

[13] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, et al. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems (MLSys)*, 2023.

[14] Shubham Tiwari, Tapan Chugh, Nash Rickert, Simon Peter, Ratul Mahajan, and Haiying Shen. Cachewise: Understanding workloads and optimizing kvcache management for efficiently serving llm coding agents. *arXiv preprint arXiv:2606.16824*, 2026.

[15] Veritas Supera. We taught our ai agents to take coffee breaks. `https://vsits.co/coffee-break-cache-keepalive/`, 2026. Accessed 2026.

[16] vLLM project. Automatic prefix caching (design document). `https://docs.vllm.ai/en/v0.9.1/design/automatic_prefix_caching.html`, 2025. Accessed 2026.

[17] Jiahao Wang et al. Kvcache cache in the wild: Characterizing and optimizing kvcache reuse at a large cloud provider. In *Proceedings of the USENIX Annual Technical Conference (ATC)*, 2025.

[18] Brandon Wie. Anthropic prompt cache ttl and cost mechanics. `https://brandonwie.dev/posts/anthropic-prompt-cache-ttl`, 2026. Accessed 2026.

[19] Zhen Zheng et al. Batchllm: Optimizing large batched llm inference with global prefix sharing and throughput-oriented token batching. *arXiv preprint arXiv:2412.03594*, 2024.