Energy Characterization of KV Cache Offloading Under Agentic Workloads

GILBERT L. MAO, Georgia Institute of Technology, USA
RUIYANG ZHOU, Georgia Institute of Technology, USA

Large Language Models (LLMs) serving complex, multi-turn agentic workloads face acute pressure on the Key-Value (KV) cache, the primary bottleneck for both GPU memory capacity and per-token energy. Modern inference engines expose KV cache offloading as a production feature, increasingly adopted for its throughput benefits, yet its energy implications remain unmeasured. We present an empirical energy characterization using per-endpoint production agentic trace replay from Applied Compute [11]. We find offloading's benefit is gated by a 95th-percentile KV-utilization threshold: above it, offloading simultaneously saves marginal GPU energy and improves throughput while below it savings fall within measurement noise. The benefit arises because offloading prevents a "recomputation tax"—under pressure, the share of prefill compute spent on recomputation rises over the low-pressure baseline and offloading yields a net reduction in energy per token (either by dropping raw GPU power or increasing throughput proportionally) as less total work is performed. The CPU and DRAM host energy of offloading stays within measurement noise, so the savings are not merely relocated to the host. In our configuration, ~32 GiB is the minimum buffer size for full benefit; we document a throughput-collapse failure mode at extreme pressure with large buffers and provide an operational decision framework.

CCS Concepts: • **Software and its engineering** → **Memory management**; • **Hardware** → **Power and energy**; *Hardware caches*; • **Computing methodologies** → *Natural language generation.*

Additional Key Words and Phrases: Large Language Models, KV Cache Offloading, Energy Efficiency, Agentic Workloads, GPU Memory Management

1 Introduction

As model parameters and context windows scale, serving infrastructure is bottlenecked not only by compute availability but by rack-level power caps and thermal limits, making energy-per-generated-token an operational metric alongside latency and throughput. The Key-Value (KV) cache stores intermediate attention states for all active requests and grows linearly with context length. Under agentic workloads that increasingly dominate production traffic—multi-turn sessions with tool calls, accumulated context windows, and concurrent long-lived sessions [1]—KV cache pressure is acute and sustained.

The GPU reserves a fixed pool of memory for the KV cache; when concurrent long-context requests fill that pool, the server must evict requests and later recompute their context at full power, whereas offloading instead copies cached state to host memory so it can be reloaded. Whether offloading helps therefore hinges not on how much GPU memory is reserved, but on how full that pool grows at runtime (Figure 1). The energy consequences of this pressure flow through two phases of LLM inference with distinct profiles. **Prefill** processes all input tokens in parallel and is compute-bound: tensor cores saturate and peak prefill power reaches the H200's 700 W thermal design limit. **Decode** generates tokens autoregressively and is memory-bandwidth-bound, drawing lower sustained power. When concurrent requests exhaust GPU KV capacity, the scheduler preempts a running request by dropping its GPU KV blocks; resumption then requires full re-prefill of the entire accumulated context at peak GPU power. KV cache offloading proactively copies every newly-computed block to host memory via PCIe (the immediate offload policy used in vLLM 0.11.0+) so a preempted request can be reloaded rather than recomputed. vLLM V1 hardcodes preemption to recompute mode (the historical preemption-swap path was removed in the V1 rewrite), making the ON-vs-OFF comparison a clean contrast between GPU-only and GPU+CPU tiered memory.

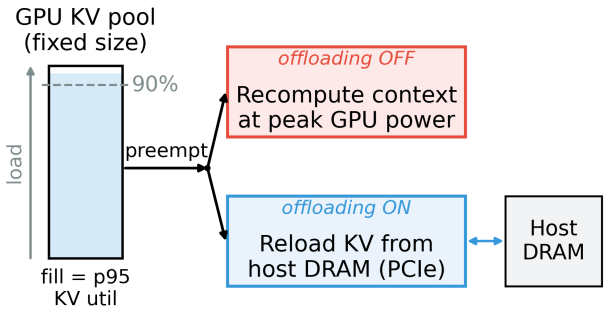

Fig. 1. The GPU reserves a *fixed* pool for the KV cache; how full it grows at runtime rises with concurrency and context length, independently of the fixed reservation. When utilization crosses a pressure threshold, the scheduler preempts a request: with offloading **off** it recomputes the request's full context at peak GPU power, whereas with offloading **on** it reloads the cached state from host DRAM over PCIe. Offloading thus sits idle below the threshold ($\Delta E \approx 0$).

While KV offloading is widely deployed to maximize throughput and prevent out-of-memory errors [5, 7, 14], its energy impact remains empirically untested. Three considerations argue for direct measurement. First, the offloading mechanism imposes *continuous* overhead. Second, the net energy balance depends on whether copied blocks actually get reloaded before they are evicted from the CPU buffer. Third, the magnitude of any savings depends on how frequently preemption fires in the first place, which is entirely workload-dependent. Two reload paths—preemption recovery and prefix-cache extension across multi-turn requests—share a single counter (`external_kv_transfer`) and cannot be decomposed post-hoc; we therefore measure aggregate ΔE across both.

We provide that measurement, replaying per-endpoint production agentic traces from Applied Compute [11] on NVIDIA H200 GPUs running Llama-3.1-8B and measuring GPU, CPU, and DRAM energy; within-node ON/OFF pairing isolates the offloading-mechanism effect from hardware variation. Our paper makes the following contributions:

Authors' Contact Information: Gilbert L. Mao, gmao8@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA; Ruiyang Zhou, rzhou343@gatech.edu, Georgia Institute of Technology, Atlanta, Georgia, USA.

(1) **Energy characterization under production agentic traces.** We find offloading's benefit is governed by a *sharp activation threshold* in p95 KV utilization—∼90% in our configuration: above it, offloading delivers significant energy savings with simultaneous throughput improvement; below it, savings fall within measurement noise on modern GQA architectures.
(2) **Operator decision framework.** We translate the threshold and the buffer plateau into a deployable rule with monitoring signals operators can query in production.
(3) **Buffer-induced preemption suppression.** We identify and characterize a counter-intuitive interaction in which larger offload buffers *increase* risk of scheduler deadlock by absorbing the eviction pressure that would otherwise trigger preemption, and translate this into a buffer-sizing rule operators can apply.

2 Related Work

LLM Inference Energy Measurement. A growing body of work measures the energy cost of LLM inference, but treats memory management as fixed background context rather than the experimental variable of interest. Stojkovic et al. [16] characterize the energy-latency-throughput tradeoff surface of LLM serving, while TokenPowerBench [10] separates prefill and decode power across model scales. Larger empirical studies develop predictive energy models across models and hardware [2], and workload-driven energy models for heterogeneous serving [17]. These works establish rigorous measurement methodologies and confirm that inference energy is a first-class systems concern, but they do not isolate KV offloading as the independent variable, or characterize the regime in which it materially shifts energy consumption.

LLM Serving and KV Cache Management. Modern LLM serving faces distinct memory pressure from variable-length, autoregressive generation. vLLM [7] introduced PagedAttention to mitigate KV cache fragmentation and exposes native KV cache offloading via LMCache-backed CPU/disk paths in its production stack. FlexGen [14] aggregates GPU, CPU, and disk memory across tiers to maximize throughput for offline batch generation; CachedAttention [3] implements hierarchical placement and scheduler-aware prefetching to reuse KV blocks across multi-turn conversations. These works optimize throughput and latency; they do not directly measure the energy cost of offloading and DMA transfer itself.

CPU Offloading in Online LLM Inference. NEO [5] is the closest system-level precedent. It offloads part of attention compute and KV cache state from GPU to CPU using asymmetric GPU-CPU pipelining and load-aware scheduling, increasing throughput while maintaining latency. Our work differs in scope: we do not propose a new offloading system or scheduler; we measure the energy consequences of an existing CPU-offloading mechanism under production agentic trace replay.

Empirical Characterization of Offloading. Empirical characterization of offloading mechanisms is an active direction. CHEOPS [13] studies the I/O behavior of offloading LLM models and KV caches to NVMe SSDs, explicitly framing the goal as characterizing framework capabilities and storage behavior. Meng et al. [9] derive a bandwidth-bounded transition point for KV offloading and report that offloaded serving can spend 99% of latency on transfers while GPUs consume only 28% of rated TDP. That work is complementary to ours: it characterizes the bandwidth-latency regime, whereas we characterize the preemption-driven energy regime under production agentic pressure.

3 Methodology

3.1 Setup

All experiments run on NVIDIA H200 GPUs (141 GiB HBM3e, PCIe Gen5 ×16) in dual-socket Intel Xeon Platinum 8562Y+ hosts with Intel RAPL energy counters available across both package and DRAM domains. We run experiments across four physical H200 nodes; all ON/OFF comparisons are within-node pairs, isolating the offloading-mechanism effect from hardware variation. To enforce strict state isolation, we restart the vLLM server for every experiment (no carryover of KV cache, prefix cache, or offload buffer state). Each experiment establishes a 30 s idle baseline, primes the caches during a warmup phase, and captures a 300–600 s active measurement window, followed by a 45 s thermal cooldown. Run order is globally shuffled (fixed seed) to randomize thermal drift across treatments.

We serve inference with vLLM 0.17.0. The treatment variable is `–kv-offloading-size` B (present for ON, absent for OFF). The model used is Meta-Llama-3.1-8B-Instruct (GQA) and we set `–gpu-memory-utilization=0.22` with 256 GiB host RAM allocated; a pressure-equivalence control uses `0.30`. This places an 8B model in the high-KV-pressure regime that 70B+ models or 128k+ contexts naturally produce on production hardware. Decoding is greedy (`temperature=0`) so output token counts are identical for matched prompts across ON/OFF. Our evaluation is based on stable execution traces; runs exhibiting framework crashes or counter-coherence failures were discarded.

3.2 Workload and Trace Replay

We replay per-endpoint production multi-turn agent traces from Applied Compute [11]: `office_work_8k` (median 37 turns, peak context 51,016 tokens, total tool-call latency 35.5 s) and `agentic_coding_8k` (median 13 turns, peak context 20,324 tokens, total tool-call latency 26.7 s). Each trace specifies initial prompt length, per-turn assistant and tool output lengths, and per-turn tool-call latencies preserved from production timing. Traces are filtered to peak context ≤ 98,000 tokens; ∼8,000 remain in each set.

Since traces contain token counts but no text (the underlying text is withheld for privacy), we generate synthetic content from the model's vocabulary at the specified per-turn lengths, prefixed by a shared 1,024-token system prompt to simulate cross-session prefix sharing universal in production deployments. All metrics use vLLM's server-reported `prompt_tokens` and `completion_tokens`. Context accumulates naturally across turns: turn N+1's prompt concatenates turn N's prompt, the generated response, and the synthetic tool output, so the prefix cache observes the same repeated structure as in production. Output lengths are forced exactly via `ignore_eos=True` and `max_tokens`. Concurrency is implemented as N async workers picking from a shuffled circular queue. We sweep concurrency c (concurrent requests) from light load ($c = 2$)

to heavy concurrent serving ($c = 6$) to characterize the pressure-response relationship for each workload. This sweep is performed at GPU memory limits of 0.22 and a 0.30 pressure-equivalence control. For offloading conditions, we sweep buffer capacities from 8 to 128 GiB, each replicated three times.

3.3 Energy Measurement

GPU energy is read from the hardware-accumulated counter `nvmlDeviceGetTotalEnergyConsumption` at window boundaries. CPU and DRAM energy are read from Intel RAPL counters [6] at window boundaries. A per-experiment 30 s idle baseline (model loaded, no inference) captures current thermal state and is subtracted. Our primary metric is marginal GPU energy per output token:

$$\text{Marginal J/token} = \frac{E_{\text{GPU}} - P_{\text{idle}} \cdot \Delta t}{N_{\text{output}}} \tag{1}$$

Because greedy decoding ensures output token counts are strictly identical between matched ON/OFF pairs, this metric isolates the end-to-end efficiency of the generation process; the resulting relative $\Delta E\%$ is mathematically invariant to whether output tokens or total tokens are used as the denominator. CPU and DRAM energy costs of the offload mechanism are recorded via RAPL and shown to be negligible in §4.5 (host ΔE within measurement noise, <1% of the GPU saving).

4 Results

We define

$$\Delta E\% = \frac{\text{marginal J/tok}_{\text{OFF}} - \text{marginal J/tok}_{\text{ON}}}{\text{marginal J/tok}_{\text{OFF}}} \times 100 \tag{2}$$

where positive values indicate offloading saves energy.

4.1 Energy Savings Scale with KV Pressure

Table 1. Energy and throughput by operating condition (buf=64 GiB unless noted; office $c = 4$ reports buf=32 only, n=3). KV p95 = 95th-percentile KV cache utilization in OFF condition; "Preempt" = preemption events per measurement window in OFF.

Condition	KV p95	Preempt	$\Delta E\%$	ΔTPS%	n
Office c=2	81.9%	0	+2.0 ± 0.6	−1.3 ± 1.4	3
Coding c=4	78.8%	0.3	+2.1 ± 4.8	+1.2 ± 1.1	3
Control c=4	81.1%	0	+2.5 ± 0.8	+0.4 ± 0.2	3
Office c=3	87.4%	0.3	+3.9 ± 5.5	+0.5 ± 1.4	3
Coding c=5	90.7%	13	+15.3 ± 0.3	+10.6 ± 0.4	3
Control c=5	91.8%	17	+14.4 ± 1.3	+18.9 ± 2.0	3
Office c=4*	92.5%	40	+26.5 ± 6.2	+16.9 ± 10.7	3
Control c=6	95.2%	53	+22.4 ± 0.8	+11.1 ± 0.6	3
Coding c=6	93.8%	55	+31.9 ± 0.9	+32.3 ± 1.4	3

± values are standard errors, propagated from the n=3 OFF and ON means as $\sqrt{\text{SE}_{\text{OFF}}^2 + \text{SE}_{\text{ON}}^2}/\bar{x}_{\text{OFF}}$.

Table 1 presents results across nine stable operating conditions at buffer ≥ 32 GiB. Here, p95 KV utilization—the 95th-percentile fill of the reserved KV pool during the run—measures the memory pressure a workload actually generates and is the runtime signal that determines whether offloading engages; it rises with concurrency and context length. The data exhibits a sharp regime boundary at a p95 KV utilization of 90% (Figure 2). Under low-to-moderate

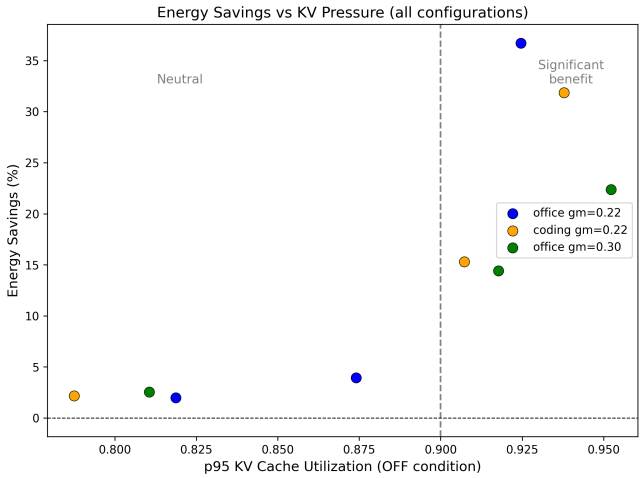

Fig. 2. $\Delta E\%$ vs. p95 KV cache utilization. A sharp threshold (~90% in our configuration) separates the within-noise regime from 14–32% savings.

memory pressure (KV p95 < 90%), offloading savings fall within measurement noise given per-experiment idle-power variation of ±2–3 W (2.0–3.9%). Conversely, in the high-pressure regime (KV p95 > 90%), offloading consistently saves 14.4–31.9%. The threshold cleanly separates all nine conditions with zero violations; with observations at 87.4% and 90.7% flanking the boundary, its precise location lies in that ~3-percentage-point gap.

The mechanistic predictor is **preemption frequency in the OFF condition**. All conditions with KV p95 >90% experience ≥ 13 preemptions per measurement window and show ΔE > 14%; all conditions below experience < 1 preemption and show ΔE < 4%. When the cache saturates at the 95th percentile, the scheduler is forced to preempt running requests to free blocks, and those preempted requests must recompute their full accumulated context at peak GPU power. Offloading prevents this recomputation.

Office concurrency 4 shows the largest energy savings (+26.5% at buf=32, n=3). At buf ≥ 64, six of nine ON experiments at this concurrency entered a throughput-collapse failure mode (§4.6) and are analyzed separately; we therefore use the stable buf=32 result, which is also the buffer carried through Table 2 for $c = 4$ mechanism analysis.

4.2 Buffer Size

At coding concurrency 6, $\Delta E\%$ varies within measurement noise across buffers of 32, 64, 96, and 128 GiB (31.9–34.8%, n=3 each). At 8 GiB, vLLM logs "cannot store N blocks" warnings in every experiment, and measured `external_kv_transfer` volume is <16% of that observed at buf ≥ 32 GiB across all high-pressure conditions: the buffer fills completely and the connector cannot build CPU-side inventory for future reloads, capturing only ~21% of maximum benefit. The data thus distinguish only 8 GiB (insufficient) from all larger sizes. The plateau is consistent with residency: at the measured offload write rate of 0.1–0.9 GiB/s, a 32 GiB buffer provides

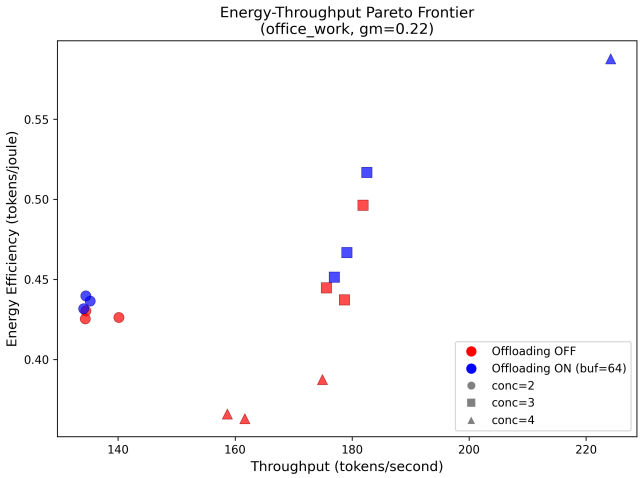

Fig. 3. Energy–throughput frontier: offloading improves both metrics simultaneously under high KV pressure.

residency exceeding the 99th-percentile tool-call latency in the Applied Compute traces (~20 s); additional capacity stores blocks that are never requested back.

4.3 Throughput is a Co-Benefit

Offloading improves throughput as well as energy at high pressure (the ΔTPS% column of Table 1, Figure 3). Both gains stem from the same cause: avoiding GPU recomputation of preempted requests. Each avoided recomputation simultaneously frees GPU compute cycles (raising tokens/second) and reduces GPU energy draw (lowering J/token). At office $c = 4$, over 21% of OFF-condition prompt tokens go to recomputation (§4.4); offloading eliminates the vast majority of this penalty. At low pressure (coding $c = 4$, office $c = 2$), throughput is unchanged within ±2%, confirming the proactive-copy mechanism imposes negligible overhead on a GQA model when the reload path is not exercised.

4.4 Mechanism: The Recomputation Tax

We examine vLLM's `prompt_tokens_by_source` counter to decompose prefill tokens into three paths: `local_compute` (fresh GPU computation at peak power), `local_cache_hit` (served from the GPU prefix cache), and `external_kv_transfer` (reloaded from the CPU buffer over PCIe).

Table 2. Token source decomposition (% of total prompt tokens), buf=64 GiB except $c = 4$ ON which is buf=32 (n=3).

	OFF			ON		
Condition	compute	cache	ext_kv	compute	cache	ext_kv
Office c=2	3.9%	96.1%	—	3.4%	96.2%	0.4%
Office c=3	5.2%	94.8%	—	3.3%	94.9%	1.8%
Office c=4	21.3%	78.7%	—	3.2%	80.3%	16.5%

At concurrency 2 (0 preemptions, no concurrent-cache competition), only 3.9% of prompt tokens require fresh computation. At concurrency 4 (40 preemptions, four concurrent contexts evicting each other's prefix blocks), this rises to 21.3%. The 17.4-percentage-point increase reflects tokens whose blocks were lost under pressure—both via preemption and via concurrent prefix-cache eviction, which share a single counter and cannot be separated post-hoc. Offloading rescues both paths via the same mechanism: CPU-buffer reload instead of full recomputation. At $c = 4$ ON (buf=32, n=3), `local_compute` drops to 3.2%, with 16.5 pp of the rescue routed to `external_kv_transfer` and 1.6 pp accumulating as additional `local_cache_hit`.[1]

GPU prefix cache hit rate appears unaffected by offloading in our measurements (±1.6 percentage points between ON and OFF); a clean orthogonality test would require disabling the GPU prefix cache while offloading is enabled, which we do not perform here.

The GPU power reduction follows directly from this avoided compute: at office $c = 4$, peak prefill power drops from 565 W (OFF, n=3) to 446 W (ON, n=3 at buf=32). The expensive recomputation passes that dominate OFF-condition GPU utilization are simply not happening.

4.5 Negligible Host-Energy Overhead

Because immediate offloading copies every KV block to host memory over PCIe, its GPU savings might be offset by CPU and DRAM energy; they are not. Computing marginal CPU and DRAM energy with the same idle subtraction as GPU, the host-side difference between ON and OFF is statistically indistinguishable from zero in every condition—pooled across the five high-pressure conditions it is -0.015 ± 0.009 J/token (n=15), with a 95% confidence upper bound below 1% of the GPU saving. The reason is a data-movement asymmetry: moving one cached KV byte over DRAM and PCIe costs ~0.2 nJ, whereas recomputing it on the GPU costs ~100 nJ—a ≥400× gap, because each KV byte embeds ~10^5 FLOPs of transformer work. Even at the measured offload rates of 0.1–0.9 GiB/s the host draws under 0.5 W of additional power, below the RAPL noise floor. Because marginal host energy is dominated by the co-located trace-replay client, this bounds the mechanism's host cost rather than measuring it precisely; either way, offloading's GPU energy savings are not relocated to the host, and the full-system effect tracks the GPU-only result. Netting host energy into the marginal metric therefore leaves Table 1's ΔE% materially unchanged.

4.6 Throughput Collapse at Extreme Pressure

At office concurrency 4 with offloading ON and buffer ≥ 64 GiB, six of nine experiments experienced complete throughput collapse: GPU power dropped to 177–182 W and output was limited to ~27,000 tokens (vs. 117–158k normally). The collapse is binary with no intermediate cases.

The discriminating feature is preemption count: every collapsed experiment recorded exactly zero preemptions. At buf=8 GiB, no collapses occurred—two of three experiments completed normally

[1] At coding concurrency 6 (buf=64), `local_compute` = 29.7% vs. 5.6% at coding $c = 4$—a 24.1 pp recomputation tax. Offloading drops $c = 6$ ON `local_compute` to 3.2%, a 26.5 pp reduction.

with 28–33 preemptions, and one experienced an unrelated server crash.

We attribute this to an engine-level synchronization defect: as large CPU buffers accumulate massive block indices under sustained pressure, asynchronous I/O offload threads block the main Python event loop (e.g., holding the GIL). This prevents the scheduler from executing its normal preemption safety valve when GPU memory physically fills. Without preemptions to forcefully pause sessions, combined demand exceeds capacity, and the scheduler deadlocks. The zero-preemption discriminator supports this hypothesis, though we have not independently isolated the thread trace.

The failure requires three simultaneous conditions: (a) per-session peak context large enough that combined demand exceeds GPU capacity; (b) buffer large enough to trigger the offload-thread blocking bug; (c) all sessions reaching peak context simultaneously. It is absent from shorter-context workloads (coding), larger KV budgets (gm=0.30), and small buffers (which overflow and force preemptions).

We observe this collapse reproducibly and diagnose its mechanism as buffer-induced preemption suppression. Because a deadlock is a scheduler-level defect—correct behavior would preempt to free blocks rather than stall—the durable remedy is an engine-level fix, not a workload change; on affected engine versions, capping the offload buffer is an interim mitigation that preserves the preemption safety valve (§5.1). Our contribution is this diagnostic mechanism, which turns an opaque throughput collapse into a monitorable failure mode.

5 Discussion

5.1 Operator Recommendations

Our results yield the following decision framework for production operators. While the specific constants are tied to our hardware configuration, the structural rules are broadly applicable:

(1) **Locate the activation threshold, then enable offloading above it.** A sharp threshold separates a regime where offloading is inert from one where it delivers large savings; *that a threshold exists is the portable claim*, not its exact location. In our configuration, this occurs at ∼90% p95 utilization, above which offloading delivers 14–32% energy savings with 11–32% higher throughput. Operationally, once the threshold T is located, alert when it is crossed, e.g. `quantile_over_time(0.95, vllm:kv_cache_usage_perc[5m])` > T (∼0.90 in our configuration).

(2) **Size the buffer to exceed tool-call latency residency.** As demonstrated in §4.2, sizing the buffer to hold blocks significantly longer than the workload's 99th-percentile tool-call latency captures >90% of the maximum possible energy savings. Undersized buffers (e.g., 8 GiB in our tests) overflow constantly and capture a fraction of the benefit.

(3) **On engine versions prone to the collapse (§4.6), cap the buffer as an interim safeguard.** The deadlock is an engine-level defect, not a property of offloading. Until a patched engine is deployed, a buffer small enough to overflow under sustained pressure (32,GiB in our configuration) keeps the preemption safety valve active.

(4) **Monitor total preemptions as the leading indicator.** A sustained nonzero preemption rate signals offloading is actively delivering benefit—in our configuration, ≥0.5/minute marks the onset, though the rate that yields a *measurable* saving scales with recomputed context length and the idle-power floor. A sustained rate of zero preemptions with offloading ON, combined with near-zero throughput and low GPU power, is the signature of the collapse failure mode.

5.2 Beyond Agentic Workloads

Our measurements are keyed to the KV-pressure regime rather than to agentic workloads per se: the recomputation tax we exploit is engaged by sustained preemption, a runtime condition any sufficiently long-context, high-concurrency workload can reach. This regime-keying mirrors prior work that decouples the offloading bottleneck from workload type via a runtime cached-to-prefill ratio [9], and the broad finding that short, low-reuse, decode-bound requests gain little from offloading [8, 15]. We therefore expect the operator rule—trigger on KV utilization and preemption rate—to transfer, while its magnitude rescales with recomputed context length: workloads dominated by long accumulated prefixes (RAG, document-QA, long multi-turn conversation) stand to save more, short single-turn chat less [3, 15]. One caveat bounds this projection: our agentic traces already exercise within-session prefix reuse and a shared system prompt, and our ΔE aggregates both the preemption-recovery and prefix-reuse reload paths, so workloads with heavier cross-request reuse may benefit more than our preemption-driven numbers suggest. Crucially, no prior work measures offloading energy directly; these projections rest on mechanism plus adjacent latency and throughput evidence, and warrant direct validation on non-agentic traces.

5.3 Limitations

Deployment regime and hardware scope. We constrain GPU memory utilization to 0.22 to place an 8B model in the high-KV-pressure regime where offloading activates. Production deployments reach this regime via different routes: larger models (70B+ on a single GPU), longer contexts (128k+), or higher tenant density. Our characterization applies to the high-pressure *regime*—sustained KV utilization above the activation threshold (∼90% in our configuration)—regardless of how it is reached; the relative ΔE should transfer to larger models since both transfer cost and recomputation cost scale proportionally with per-token KV size, though multi-GPU tensor parallelism would introduce inter-GPU communication energy not measured here. All experiments use vLLM V1 in its default colocated prefill/decode mode on H200; prefill-decode disaggregation [12, 18] would alter KV pressure dynamics. Threshold behavior and direction of savings replicate across all nodes tested.

vLLM configuration and workload realism. We disable async scheduling symmetrically in both conditions; preemption frequency—our headline mechanistic predictor—may shift under default async-scheduled deployments where scheduling decisions are not aligned to batch boundaries. Our N-worker trace replay holds concurrency constant, whereas production deployments operate open-loop with autoscaling-mediated load balancing; the sustained KV pressure

observed at fixed concurrency may be transiently relieved by production capacity adjustments. Synthetic token content preserves within-trace structural similarity for prefix caching but does not reproduce cross-trace structural similarity (shared envelopes, common error formats); the direction of this bias on ΔE is ambiguous, as real cross-trace reuse would benefit both ON and OFF conditions.

KV Cache Quantization. Our baseline evaluates BF16 KV caching. While 8-bit (FP8/INT8) KV caching is increasingly common in production deployments [4] and halves the memory footprint per token, it concurrently halves the PCIe transfer cost. Consequently, FP8 would push the ∼90% activation threshold to a higher concurrency level or longer context length, but the fundamental energy tradeoff (cheap PCIe transfer vs. expensive GPU recomputation) remains.

Model architecture and content-dependent execution. Because the traces carry token counts but not text, we synthesize token content—valid here because a dense GQA model's per-token compute and KV footprint depend only on token *counts*, which we replay exactly, not on token *values*. This insulation breaks for architectures whose per-token execution is content-dependent. Mixture-of-Experts routing is the clearest case: the experts a token activates depend on its content, so synthetic tokens would mis-estimate both per-token energy and the hot-expert working set. Speculative decoding is analogous—real text is more predictable than random tokens, so synthetic content would understate draft acceptance and overstate compute—as are early-exit and mixture-of-depths models, whose per-token depth varies with content. For these architectures, our energy figures are indicative only; faithful measurement requires real trace content and is left to an extended study.

Statistical power. Each condition comprises n=3 paired runs. The regime separation is robust—high-pressure savings exceed their standard error by 15–30×—but per-condition standard errors near the threshold are imprecise, and for the low-pressure rows they are large relative to a mean effect that is within measurement noise by construction. Additional repeats would tighten these near-threshold estimates without affecting the headline separation.

6 Conclusion

KV cache offloading was introduced as a throughput optimization, yet our measurements suggest it is also one of the few software-only energy levers available to operators of existing serving infrastructure. The lever engages above a sharp KV pressure threshold that production agentic workloads now reach routinely—and that single-shot benchmarks miss entirely. Its two-sided behavior, with energy savings under controlled overflow and deadlock under large buffers, means the operational rule is more nuanced than a binary toggle. As agentic traffic grows toward its projected share of inference, representative workload selection may matter as much for energy claims as model architecture or hardware generation.

Acknowledgments

This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.

References

[1] Anthropic. 2026. Anthropic Economic Index: Understanding AI's Effects on the Economy. https://www.anthropic.com/research. Accessed: 2026-05-19.

[2] Francisco Caravaca, Angel Cuevas, and Ruben Cuevas. 2025. From Prompts to Power: Measuring the Energy Footprint of LLM Inference. *arXiv preprint arXiv:2511.05597* (2025). https://arxiv.org/abs/2511.05597

[3] Bin Gao, Zhaoyu He, Purushottam Sharma, Qixuan Kang, Djordje Jevdjic, Junda Deng, Xinhao Yang, Zehuan Yu, and Peng Zuo. 2024. Cost-Efficient Large Language Model Serving for Multi-turn Conversations with CachedAttention. In *2024 USENIX Annual Technical Conference (USENIX ATC '24)*. USENIX Association, 1117–1132. https://www.usenix.org/conference/atc24/presentation/gao-bin

[4] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2025. KVQuant: Towards 10 Million Context Length LLM Inference with KV Cache Quantization. arXiv:2401.18079 [cs.LG] https://arxiv.org/abs/2401.18079

[5] Xuanlin Jiang, Yang Zhou, Shiyi Cao, Ion Stoica, and Minlan Yu. 2024. NEO: Saving GPU Memory Crisis with CPU Offloading for Online LLM Inference. arXiv:2411.01142 [cs.DC] https://arxiv.org/abs/2411.01142

[6] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems* 3, 2 (2018), 1–26.

[7] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th ACM Symposium on Operating Systems Principles (SOSP '23)*. ACM, 611–626. doi:10.1145/3600006.3613165

[8] Yuhan Liu, Yihua Cheng, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Rui Zhang, Kuntai Du, and Junchen Jiang. 2025. LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference. arXiv:2510.09665 [cs.LG] https://arxiv.org/abs/2510.09665

[9] William Meng, Benjamin Lee, and Hong Wang. 2025. Understanding Bottlenecks for Efficiently Serving LLM Inference With KV Offloading. arXiv:2601.19910 [cs.AR] https://arxiv.org/abs/2601.19910

[10] Chenxu Niu, Wei Zhang, Jie Li, Yongjian Zhao, Tongyang Wang, Xi Wang, and Yong Chen. 2026. TokenPowerBench: Benchmarking the Power Consumption of LLM Inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 32582–32590. doi:10.1609/aaai.v40i38.40535

[11] Oam Patel and Linden Li. 2026. *Production Multi-Turn Agent Traces from the Applied Compute Fleet.* Technical Report. Applied Compute.

[12] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. 2024. Splitwise: Efficient Generative LLM Inference Using Phase Splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA '24)*. IEEE, 400–414. doi:10.1109/ISCA59077.2024.00039

[13] Zebin Ren, Krijn Doekemeijer, Tiziano De Matteis, Christian Pinto, Radu Stoica, and Animesh Trivedi. 2025. An I/O Characterizing Study of Offloading LLM Models and KV Caches to NVMe SSD *(CHEOPS '25)*. Association for Computing Machinery, New York, NY, USA, 11 pages. doi:10.1145/3719330.3721230

[14] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. FlexGen: High-throughput Generative Inference of Large Language Models with a Single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*. PMLR, 31094–31116. https://proceedings.mlr.press/v202/sheng23a.html

[15] Vikranth Srivatsa, Zijian He, Reyna Abhyankar, Dongming Li, and Yiying Zhang. 2024. Preble: Efficient Distributed Prompt Scheduling for LLM Serving. arXiv:2407.00023 [cs.DC] https://arxiv.org/abs/2407.00023

[16] Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri, and Josep Torrellas. 2024. Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference. *arXiv preprint arXiv:2403.20306* (2024). https://arxiv.org/abs/2403.20306

[17] Grant Wilkins, Srinivasan Keshav, and Richard Mortier. 2024. Offline Energy-Optimal LLM Serving: Workload-Based Energy Models for LLM Inference on Heterogeneous Systems. In *Proceedings of the 3rd Workshop on Sustainable Computer Systems (HotCarbon)*. https://arxiv.org/abs/2407.04014

[18] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. 2024. DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation (OSDI '24)*. USENIX Association. https://arxiv.org/abs/2401.09670