THE FIXED COST IS WORTH CUTTING: END-TO-END SPEEDUPS FOR ACTIVATION-SPARSE LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

The roughly 90% of FFN intermediate rows that sparsity-trained language models leave dead at inference time cannot be skipped for free: conditional-skipping mechanisms pay costs that do not shrink with the amount they skip, and end-to-end speedups have stayed far below this byte accounting. We capture this in a compact cost model, $T(a) = F + a \cdot B/\beta$, which yields a decision rule—cut the fixed cost. Guided by this rule, we rebuild the sparse decode path around a lean row-skipping kernel, a sparsity-aware int8 recipe, and whole-step CUDA graphs. The rebuilt stack reaches 2.66× over the model as shipped and posts 1.38×, 1.55×, and 1.94× at matched graph caliber over the official CUDA-graph baseline, a state-of-the-art sparse kernel, and the strongest measured dense stack, while sitting 1.36× above the sustained-bandwidth floor. The stack is a sparsity-trained LLaMA-2-7B under single-batch decoding on one RTX 5090, with scale probed separately (Sec. 4.4). The same experiments expose a second, temporal axis of redundancy absent in dense controls: deep FFN inputs hold a cosine of at least 0.999 across adjacent steps on 46.4% of step pairs at 7B and 82.1% at 13B, orthogonal to spatial skipping and strengthening with scale. A training-free, zero-parameter reuse mechanism cuts 18.7% off the decode step on repetition-heavy decoding (a 1.23× speedup at greedy-token agreement 1.0) at zero measured cost on natural text, with its reused steps' divergence measured and bounded. On long-chain arithmetic reasoning the full stack stays within the pre-registered tolerance with a 2.0% paired flip rate, while aggressive accelerations exceed 10%. All claims rest on a pre-registered fidelity protocol (distribution KL, decision agreement, task flip). Code for the full stack will be released.

1 INTRODUCTION

1.1 BACKGROUND AND RELATED CONTEXT

Autoregressive decoding dominates LLM serving cost, and in the small-batch, single-stream regime (interactive chat, on-device inference, agent loops) it is memory-bound: each token streams the full weights from HBM once, so latency is weight bytes over bandwidth (Sec. 2.2) and cannot be amortized over batches. Two families reduce the bytes per step: quantization shrinks each byte (Sec. 5.2), and activation sparsity skips whole groups of bytes by training FFN activations to be mostly zero. ReLU-era models showed natural sparsity (≈95% on OPT (Mirzadeh et al., 2023)), exploited by predictors such as DejaVu (Liu et al., 2023) and neuron-granular systems such as PowerInfer (Song et al., 2024b). SwiGLU removed it, and the field rebuilt it in two ways. Training-free magnitude pruning (CATS Lee et al., 2024, TEAL Liu et al., 2025a, the latter reporting up to 1.53–1.8× over a graph-enabled baseline) buys the sparsity cheaply but with false negatives. Sparsity-oriented continued training (ReLU Strikes Back Mirzadeh et al., 2023, ProSparse Song et al., 2024a, TurboSparse Song et al., 2024c) recovers ReLU-family sparsity with a ground-truth gate at ≈90% dead FFN rows and no measurable accuracy loss, and the ProSparse checkpoints used here decode with 6.79% live rows. A third family cuts the bill differently: adjacent decode steps can need largely the same computation, a repetition exploited by sequence-level expert reuse (Dong et al., 2024), recycled states (Jiang et al., 2023), and cached prefixes (Gim et al., 2024).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1.2 THE GAP: PROMISE VERSUS REALIZATION

Realized end-to-end speedups remain far below what the byte accounting suggests, and no prior work systematically accounts for where the difference goes. **What absorbs activation sparsity's promised speedup?** The answer keeps apart two questions: how much compute a method removes, and how close its wall-clock time gets to the bandwidth bound.

1.3 OUR ANSWER: A FIXED-COST MODEL OF CONDITIONAL SKIPPING

We trace the gap to a *fixed-cost trap*: the folk belief that activation sparsity yields near-lossless speedups for free once a sparse kernel is in place holds only while the reduction is partial, and it breaks once the full stack is engaged. Conditional-skipping mechanisms pay costs that do not scale with what they skip. We capture the trap as $T(a) = F + a \cdot B/\beta$, a rate-independent fixed term F plus a proportional term, and draw a decision-granularity rule: skipping pays only when the decision's fixed cost falls below the proportional savings at that granularity. Amdahl's law and roofline ceilings state what a perfect removal could save. The ledger answers the other question: what the unavoidable costs are made of, segment by segment, and which of them can be driven to zero. The ceiling says where the limit sits, the decomposition says how to approach it. Three verified predictions follow. P1: a lean row-skipping kernel that hoists detection escapes the trap at the deployment point (Sec. 4.3). P2: the proportional term is already near peak bandwidth (98.2%, Sec. 3.1), so only F is worth cutting. P3: refining granularity below the 12-layer block costs more than it saves (per-layer gate, 0.8045×, Sec. 4.5). The decomposition is an accounting frame, not a theorem, and every claim it organizes is verified under the pre-registered fidelity protocol (KL = 0, argmax agreement 1.0, long-chain flips within tolerance, Sec. 3.4).

1.4 FINDINGS AND CONTRIBUTIONS

- **The ledger and the rule read off it.** The fixed-cost trap ($T(a) = F + a \cdot B/\beta$) and the decision-granularity rule: the trap is exposed in the reference implementation and escaped by a lean row-skipping kernel at the deployment point (Secs. 3.1, 4.3, 4.5). The per-layer increment closes by attribution: replaying the gate's recorded decisions without synchronization lands on the no-gate baseline, placing that increment in the synchronous decision path (App. D.4).
- **The stack as the frame's existence proof.** The kernel, int8 recipe, and whole-step CUDA graphs compose to 1.38× over the official CUDA graph baseline, 1.55× over TEAL's graph kernel, and 1.94× over the strongest dense stack we measured, all at matching caliber, bf16 baselines against our int8 stack, with 2.66× over the official eager release form users download. The stack sits 1.36× above the bf16-dense ledger floor and lands within the pre-registered tolerance on the hardest long-chain trial we know (2.0% paired flip rate, Sec. 4.2), while aggressive accelerations break the chains.
- **A second axis, exposed by the same experiments.** In the ProSparse family (7B and 13B), deep FFN inputs hold a 0.999 cosine across adjacent steps far more often than a same-size dense model, a recipe-specific effect that strengthens with scale. Its prefix boundary is measured rather than assumed: a single prepended token collapses the stable tail at 7B with no recovery over 256 tokens, while at 13B chat-templated prefixes leave the regime intact and the collapse is BOS-specific. The regime itself is mapped across twelve models and four knobs (Sec. 4.4, App. F, App. F.6). A training-free, zero-parameter mechanism exploits it at greedy agreement 1.0 on 7B's default gate, divergence measured and bounded (1.000× on natural text, 1.230× only under sustained reuse, Sec. 5.3).

2 MOTIVATING ANALYSIS: PHENOMENA BEFORE METHOD

What the byte accounting promises, what wall-clock time delivers, and which phenomenon explains the gap. No method is proposed here.

2.1 THE BYTE PROMISE VERSUS THE SPEED REALITY

At the deployment live-row rate of 6.79% (Sec. 4.1), dead rows are roughly 93% of the FFN intermediate bytes, so skipping them *should* remove that fraction of FFN compute. The realized

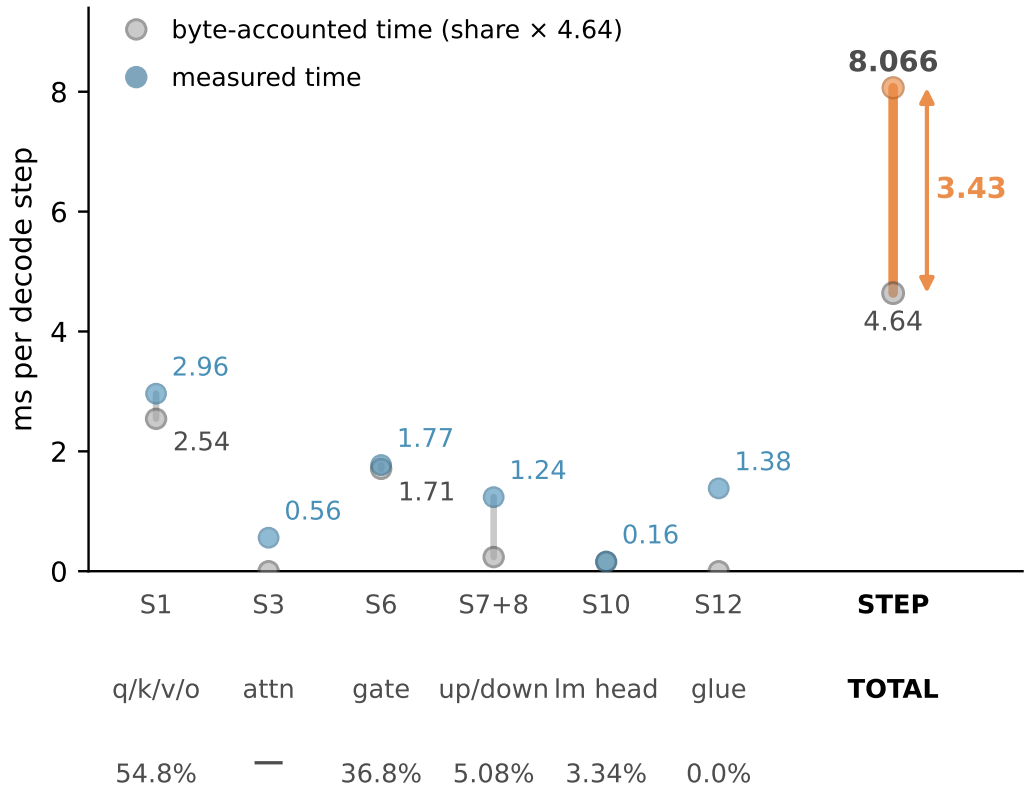

Figure 1: One decode step, two readings, one coordinate system. Each segment column carries two markers: gray = budgeted time (byte share × 4.64 ms), blue = measured, summing to 8.072, 0.07% apart from the independent total. The STEP TOTAL dumbbell marks the core tension: 8.066 ms measured against 4.64 ms budgeted, a 3.43 ms (42.5%) fixed-cost gap.

speedup is a fraction of the promise: the official release documents about 1.5× from its sparse kernel, and our reproduction reaches 1.20–1.45× over its own dense baseline (Sec. 4.3), a gap not unique to one implementation.

2.2 The whole-step timeline ledger

One decode step of ProSparse-LLaMA-2-7B (bf16, batch=1, graph caliber, our engine) is 8.066 ms. The step's weight traffic is 7.84 GB: against the 1,689.8 GB/s sustained band the floor is **4.64 ms**, and the step realizes 57.5% of sustained bandwidth. The ledger is recomputable from public dimensions alone (App. B). **How to read it (Fig. 1):** ask which segment a speedup touches and its byte share: a 2× win on S7+S8 (the up/down projections, 5.08% of bytes) moves only ≈15% of the step, so kernel-level and end-to-end numbers are reported separately. And the ledger's second reading: the reference kernel spends a large, nearly constant fraction of its time regardless of how many rows are live, extrapolating to ≈80% of dense time even at 0% live rows (Fig. 3, App. B)—the removed bytes are proportional, yet the time does not follow them down, so something rate-independent is paid at every rate.

The two readings pose one question: if the saved bytes are proportional but the saved time is not, the missing time is a fixed cost that scaling cannot remove (Sec. 3).

3 METHOD: FROM THE FIXED-COST GAP TO A SECOND AXIS

Four blocks: the cost model (T(a)=F+a·B/β) with its measured inputs, the analysis of why the trap holds, the temporal redundancy hypothesis with its phenomenon test, and the stack it dictates.

3.1 COST ACCOUNTING AND DESIGN RULES

batch=1 decoding is a byte game: segment time ≈ segment bytes ÷ bandwidth, anchored at 1,689.8 GB/s sustained on an RTX 5090 (large-buffer reads, with readings above 1,705.1 GB/s discarded as L2-resident). Eager and CUDA graph execution differ by 2.4–3.0× in launch tax (eager pays ≈5–10 μs per kernel, hundreds per step), so **all end-to-end numbers except the explicitly labeled eager-caliber denominators (L0 and the `torch.where`-fixed dense, Sec. 4.3) are graph caliber**, with no cross-caliber arithmetic outside those disclosed comparisons.

The fixed-cost model and the decision-granularity rule. Write a skippable segment's time at live-row rate a as T(a) = F + a·B/β: a rate-independent fixed term F (detection, reduction, launch) plus a proportional term. **Sparsity's promise is that a·B/β vanishes linearly with a, yet F does not.** The proportional side is capped by $\beta/\beta_{\text{peak}}$, measured at 98.2% on our segments (L0, soft, anchored at 1,659 / 1,689.8 GB/s, with the decomposition below), so **only F is left to cut** (the recipe can still shrink bytes, a separate axis). Any conditional-skipping mechanism at granularity g carries a fixed decision cost D(g) independent of what it skips, and **skipping pays only when D(g) < the proportional savings at granularity g**, verified on row and layer skipping (Sec. 4.5). Structurally akin to Amdahl's serial-fraction framing, its extension to MoE routing carries a minimal measured instance (Sec. 6.2, App. B.14), while early-exit and speculation remain untested. Method note: every limit claim must name its anchor, its layer, and its kind—mistaking an L0 value for an L1 limit is the most common source of phantom ceilings (App. C).

The model's two inputs are measured, not assumed. The official kernel's T(a) reads nearly flat on a live-row-rate sweep from 0% to 90% (Fig. 3, App. B), extrapolating to F ≈ 0.80·T(1), while ours falls at a proportional slope. Ablating the up/down segment puts that slope at 1,659 GB/s, 98.2% of the 1,689.8 GB/s anchor, so the proportional side is full and only F is left to cut: split-K never completed on this segment, and access-pattern tunings stay below the anchor.

The sparsity×quantization sensitivity rule. Recipes tuned on dense models transfer poorly to activation-sparse ones for three reasons: sparsification contracts the signal onto few live rows with heavy tails that int4's 16 levels under-resolve, rotations remix dead rows' exact zeros with live outliers and destroy the zero-false-negative structure our gate relies on, and σ-clipping removes the outliers that sparse activations concentrate on. **int8** is the safe line (13.5% vs. RTN's 11.0%, Sec. 4.2). Probes (layers 16/24/28, 4,096 GSM8K tokens, dense control) confirm both: native-basis skips of exact-zero rows are exact (0.0), the same skip under a Hadamard-128 rotation incurs 34.4–48.1%, and 3σ clipping costs the sparse model a median 0.841 vs. 0.531 for dense. The same distribution explains why dense recipes do not transfer: the gap tracks the bit-width's distribution rather than recipe naivety (full matrix: Sec. 4.2).

3.2 ANALYSIS: WHY THE TRAP HOLDS

Why the official kernel carries the fixed cost. Each thread block **scans all 11,008 intermediate rows to test liveness**, reads the live rows' weights, and performs a cross-warp/block reduction: the scan is O(intermediate dimension) *regardless of how many rows are live*, and the reduction round-trips are rate-independent. Detection can be hoisted out of the kernel, since the liveness information **already exists** (the gate projection is computed anyway as the FFN's first half).

3.3 THE TEMPORAL REDUNDANCY HYPOTHESIS

The temporal redundancy hypothesis. Sparsity training collapses each step onto a small live subnetwork whose input drifts slowly, so perturbations that do not change the live set leave the output almost unchanged: **the discreteness of the live/dead structure is a plausible source of temporal reusability (synthesized in Sec. 6.1).** Falsifiable prediction: a dense model should show ≈no adjacent-step pair clearing the 0.999 cosine bar. **Four questions: which bytes does it save? By what information does it decide? What does that information cost? How is fidelity proven?**

Existing methods separate cleanly (Sec. 5), and our temporal reuse uses *the previous step's outputs*, already resident on the device, at ≈zero tax (classification: App. C). Sec. 4 verifies the cut-F design and the sensitivity rule, with the per-layer-gate negative result in Sec. 4.5.

The prediction holds, with the phenomenon mapped. The sparse deep band (L20–31) reaches cosine ≥0.999 on 46.4% of adjacent-step pairs (the L20–31 average under the 200-segment protocol, App. F), above 50% in L26–31, and the exceedance rate decays monotonically across the threshold ladder (62.7% → 58.9% → 46.4% → 25.4% → 0.05%), with the operating gate (0.9995) inside this set (Fig. 9). A same-size dense model measures zero exceedances at every threshold (0/19,000 pairs per layer, a measured zero at ≤0.016%, ≈2,900× below the sparse rate)—the falsifiable outcome the hypothesis names, not a floor of the metric. Sec. 6.1 reads this asymmetry as the migration of redundancy, a second redundancy orthogonal to spatial skipping.

3.4 Concrete execution

The stack (detail: App. A.9). Four components, multiplicative, none new in isolation, and ordered by the accounting frame. **The lean row-skipping kernel** (S7+S8: 1.236 ms, 5.08% of bytes) hoists detection to the ground-truth gate (zero false negatives, one gate value per row), applies the live predicate to the weight loads so dead rows contribute no bytes and reads stay block-aligned, fixes the reduction count, and replays within tolerance before timing. **The lagged-cosine dual-graph mechanism** (deep FFN) decides in-graph and asynchronously: each step writes the 12 deep layers' (L20–31) post-LN2 hidden states into a static buffer and computes cosine(x_{t−1}, x_{t−2}), the host reads 12 floats back and opens for the next step only if all layers exceed 0.9995, and each step replays graph A (refresh) or graph B (skip the deep FFN, add the cached outputs), chosen from the *previous* step's signal, with a gate failure returning to A so reuse always draws on the most recent *true* values. **The sparsity-aware int8 recipe** (S1: 2.962 ms, 54.8% of bytes): W8A16, no rotation, integer zero-point asymmetric, g128 granularity (per-tensor scales are biased by dead-row zeros), served by a custom W8A16 GEMV. **Composition:** whole-step CUDA graph (cuts S12 launch tax) → S1 int8 → lean row-skip → temporal reuse.

The fidelity protocol and zero-trust verification (full text: App. B). Four pre-registered layers: truncated KL percentiles, argmax decisions, replay checks, and paired long-chain flips. **The zero-false-negative contract:** the kernel's skipped set equals the gate's ground-truth set, zero false negatives, at the price of the gate tax (36.8% of bytes). Every number was independently recomputed from raw records (one caliber trap caught: App. C.8), and portability is in the protocol (App. G).

- **Caliber sensitivity, disclosed.** The small-set PPL failure (+6.24%, failing our <1% criterion) and the originally pre-registered ΔNLL gate miss (0.00995) are recorded rather than selecting gates post hoc, while on a 133k-token set ΔNLL = −0.00405 (CI [−0.026, +0.018]), KL, and the long-chain flip (2.0%) all pass: **the claim rests on the large-set triple of ΔNLL CI, KL, and task-level outcomes** (full disclosure: App. B). Recipe ablations: App. A.

4 Experiments I: Main Results

All three predictions of Sec. 1.3 close here or where they were opened: P1 in Sec. 4.3, where the lean kernel escapes the trap at the deployment point, P2 in Sec. 3.1, whose 98.2% decomposition entered the model as a design input, and P3 in Sec. 4.5, where the per-layer gate measures 0.8045×, the penalty the rule predicts.

4.1 Setup

RTX 5090 (sm_120, 96 MB L2, sustained bandwidth 1,689.8 GB/s), Torch 2.8.0+cu128 / triton 3.4.0, and ProSparse-LLaMA-2-7B (official SparseLLM weights, bf16, FATReLU threshold 0.01, deployment live-row rate 6.79%). All end-to-end numbers are at CUDA graph caliber (baseline stack 8.066 ms, independently re-measured 8.042 ms, 0.3% apart), and fidelity claims go through the Sec. 3.4 protocol. Eager timings drift ≈20% across sessions, so every ratio is single-session, except the graph-class dense comparator rung (session structure in App. A.1). The ladder re-runs on

Table 1: The end-to-end waterfall: four same-stack measured levels (App. D).

Level	Configuration	ms/token	Step	Total
L0	stock HF, official release form (eager, boolean-mask FATReLU)	16.752	—	1.00×
L1	+ our engine, bf16 (whole-step CUDA graph + lean row-skipping kernel)	8.042	2.08×	2.08×
L2	+ S1 int8 (qkvo, W8A16 g128, no rotation, integer zero-point)	7.019	1.15×	2.39×
L3	+ gate8 full recipe (gate int8 as well)	**6.291**	1.12×	**2.66**×

a second host within 1.5% (the L1/L3 ratio moves 1.278 → 1.311), and cross-caliber statements use the portable class (App. G).

4.2 Fidelity and accuracy: does the stack survive the hardest trial?

Long chains are the hardest test: GSM8K 4-shot CoT ×200, paired per question (Fig. 2, App. A).

Pre-registered criteria (<3% passes, >10% breaks, full table: App. A): int8 passes (2.0%, 4 bad / 0 good, chain completeness 0.510, bitwise-identical replay), while int4 (11.0%) and well-tuned AWQ int4 (13.5%) stay above the 10% line, whole-layer skipping (15.5%, chains 0.51→0.19) breaks, and head pruning degrades mildly (4.0%). The int4 gap is not an artifact of a naive recipe: the standard AWQ pipeline raises accuracy (0.115 vs. RTN's 0.070) while raising the flip rate by as much as RTN, a caliber trap for accuracy-only checks that the paired criterion catches (App. B).

- **Bit-width monotonicity:** int8 is near-lossless (KL 0.00863 < 0.05, ΔNLL −0.00405, 95% CI crossing 0) while int4 stays outside tolerance on every route, naive (best KL = 0.319 ≫ 0.05) or well-tuned (13.5% flip).

Resolution. V0's 16% accuracy reflects the task difficulty, so the test is paired. V1's 2.0% (4/200) reruns on the 1,319-question pool at 2.81% (17 bad / 20 good), and a noise-floor arm shows 0.0 (Wilson 95% upper bounds 5.03% and 3.84%: App. B). **Full-stack (n=200):** the release engine reproduces V0 (0.160, 0.510, 2.0%), and with int8 on top the full stack holds 2.0% (3 bad / 1 good) at acc 0.150.

4.3 Speed and acceleration: what does the stack achieve end-to-end?

Kernel-level (P1). At the deployment rate (6.79% live rows, identical inputs, event timing, graph caliber) our kernel runs **1.12–1.17**× the official ProSparse kernel and **1.27–1.97**× TEAL. The official repository's own test entry shows 1.20–1.45× over its dense path. A config-fixed TEAL rerun gives 1.29–1.95×, within 1.4%. All ratios are identical-input, with published numbers a different caliber (App. E).

The rate sweep locates the official/ours crossover at ≈8–10% live rows, with the deployment mean (6.79%) below it (App. A.2 maps the reference's only winning region).

- **Segment speed:** int8 W8A16 gives 1.465× on S1 (4 runs, range 0.27%), diluting to the L2 rung of 1.15×: the root cause was implementation efficiency (eff 0.660), not bit-width—an L0 cause cured by an L0 fix in the Sec. 3.1 taxonomy (arms: App. A.3).

End-to-end waterfall (Table 1, Fig. 6, App. D). Every level is measured by one same-stack script (no derived or interpolated numbers, product check 2.083 × 1.146 × 1.116 = 2.66, greedy outputs identical over 16 steps, max-logit relative error 0.0157):

Caliber note (full accounting: App. D). Of three labeled denominators, the primary is **matching graph caliber at 1.94**× (pure jump 1.92×, our same-caliber gain 1.08×), with the eager comparisons (2.01×, 2.66×) cross-caliber.

Floor and competitors. 7.84 GB ÷ 1,689.8 GB/s = 4.64 ms floor, so 6.291 ms sits 1.36× above the bf16-dense ledger's floor (a conservative reference: the int8 stack reads fewer bytes), the gap set by S12 non-weight overhead and S3 attention (App. D). End-to-end, gate8 is 1.38× the official kernel's graph path (8.707 ms), 1.55× TEAL's (9.765 ms), and 1.21× for our bf16 engine against that path, all with bf16 baselines against our int8 gate8 stack. Graded weight: 1.38× closest to a

shipped baseline, 1.55× same-caliber, 1.94× our strongest dense stack (engine-dense graph arm, App. A.1), 2.66× cross-caliber.

4.4 Temporal reuse: mechanism, fidelity, and speed

The phenomenon and its dense control are established in Sec. 3.3 (Fig. 9). What remains: the boundary map, and the mechanism's fidelity and speed.

Boundary map (full sweep: App. F). One prepended token collapses the stable tail at 7B, while at 13B the collapse is BOS-specific and chat-templated prefixes leave the regime intact (App. F.6). The regime tracks the ProSparse-LLaMA-2 configuration: the twelve swept models span 2.6–6.7% live rows against $\geq$14.9% for every zero-exceedance model, strengthening with scale (13B 82.1% vs. 7B 46.4%), and threshold-forcing isolates the configuration from the rate (App. C.6). Its exact zeros are functional rather than bookkeeping, the plausible carrier of the regime and the hypothesis source of Secs. 3.3 and 6.1 (rotated-basis loss: Sec. 3.1). Raising the FATReLU threshold amplifies it monotonically (7B 32.0% $\rightarrow$ 47.6%, 13B 66.5% $\rightarrow$ 98.5%) while the sensitive metric stays flat inside the usable range, and a forcing template cannot induce it (App. F).

Fidelity (Fig. 8, App. F). Under natural-text teacher forcing the mechanism posts argmax agreement 1.0 with KL median 0 and p90 0, and in the reuse-heavy regime fired-step divergence grows along the chain ($\approx$0.20 $\rightarrow$ 1.18 nats), while at the guarded 13B setting (τ = 0.9999, cap 8) coverage is 0.805 at agreement 1.0 with fired-step KL median 0.0003 (matched-probe profile and long-range behavior: App. F.5).

Speed, with its cause stated. In the reuse-heavy regime (97.9% of steps on graph B) the mechanism reaches 1.230× at graph caliber (11.17 $\rightarrow$ 9.08 ms/token, 18.7%) — a property of the model's self-reinforcing loops, not a deployment caliber — while the guarded setting returns a fidelity-clean 1.154×. On natural text it measures 1.000× (zero cost, gate-open fractions at most 2.4%), with nothing to switch off outside the regime (Fig. 7).

4.5 What the rule excludes: the granularity negative result

P3. Refining reuse granularity from a 12-layer block to per-layer decisions measures a net penalty: **13.878 ms/token vs. 11.166 ms** for the single full-compute graph, **0.8045**×, with the incremental per-layer-gate step at 1.83 ms and the skip it unlocks saving only 0.3–0.5 ms. **The measured penalty is $\approx$2.7 ms (component account and replay attribution: App. D.4).** One notch finer granularity, one notch of fixed cost: this penalty is a soft cost in the taxonomy of Sec. 3.1, which is why the 12-layer block survives it, and the same protocol rejects whole-layer skipping (V3b: 15.5% flip, chains 0.51$\rightarrow$0.19). The four questions opened in Sec. 3.3 are each answered by a negative result, every excluded direction falling to the same ledger entry (full list and table: App. C.8, App. D). GateSkip (Laitenberger et al., 2026) reaches token-wise layer skipping with fine-tuned residual gates, the other fork of the rule: per-layer information either costs $\approx$nothing (our D(g) test) or is paid for in training.

5 Related Work

The phenomena, theory, and results above establish the problem and its solution before we situate the work. batch=1 decoding accelerations fall on a 2-D map: where the saving occurs (within a step or across steps) against what is removed (whole bytes or bits per byte). The four cells: (i) per-step, bytes, activation sparsity. (ii) per-step, bits, quantization. (iii) cross-step, bytes, speculative decoding, early exit, and the cross-step reuse families (Sec. 5.3). (iv) cross-step, bits, outside our scope. Temporal reuse is cross-step, bytes: deep-FFN bytes that persist across steps, at $\approx$zero tax. The nearest prior, GRIFFIN (Dong et al., 2024), also exploits cross-token stability, but on dense models and once per sequence, while MoE routing is an orthogonal axis the rule also prices (Secs. 3.1, 6.2).

5.1 Activation sparsity: models and kernels

Making models sparse. ReLU-family models exhibit natural activation sparsity (Mirzadeh et al., 2023), and ProSparse (Song et al., 2024a) reaches $\approx$89–93% via FATReLU under progressive

regularization, TurboSparse pushing further (dReLU). Both carry a *ground-truth* gate with zero false negatives, at the price of reading the gate weights in full (36.8% of traffic, Sec. 2.2). Sparsing Law (Luo et al., 2025) quantifies the training side. Our ledger prices what realized sparsity pays at inference. **Using the sparsity.** Ground-truth-gated kernels include the official ProSparse operator (our baseline). TEAL (Liu et al., 2025a) applies training-free magnitude sparsity at 40–50% and reports up to 1.53–1.8× over a graph-enabled baseline at a different operating point. Predictors (DejaVu, CATS) save the gate tax but introduce false negatives, PowerInfer splits hot/cold neurons across GPU and CPU, and FFN-SkipLLM (Jaiswal et al., 2024) skips whole layers, orthogonal to ours (Sec. 3.2). Recipes (Zhang et al., 2024), rotated-basis routes (Liu et al., 2025b), further systems (Akhauri et al., 2024) (Zhou et al., 2024) (Shin et al., 2025b) and kernels (Cheon & Kang, 2025) (Shin et al., 2025a) all pursue the same per-step savings, and input-adaptive selection (Yang et al., 2025b).

5.2 Quantization: why dense-recipes do not transfer

GPTQ (Frantar et al., 2023), AWQ (Lin et al., 2024) (whose salient-channel protection is akin to our no-clipping rule), SmoothQuant (Xiao et al., 2023), QuIP/QuIP# (Tseng et al., 2024), QuaRot (Ashkboos et al., 2024), SpinQuant (Liu et al., 2025c), ParoQuant (Liang et al., 2026), and others rotate or reshape distributions to make them quantization-friendly. Two ingredients hurt activation-sparse models: rotations remix dead-row zeros with live outliers, breaking the zero-false-negative contract, and σ-clipping removes functional outliers (Sec. 3.1). ParoQuant corroborates our hardest-test choice (error compounds over long-CoT, and they repair int4 with Givens rotations), while on sparse models int4 stays outside tolerance even with activation-aware tuning (Sec. 4.2).

5.3 Cross-step computation reuse and system-level methods

Cross-step computation reuse. Our stability regime is bound to a 0.999 cosine bar that dense controls never clear (zero exceedances, Sec. 4.4), and its decision signal is the sparse model's own ground-truth gate, read one step late: no trained router, no adapter, no added parameters (Sec. 6.1). **The neighborhood (full list: App. C).** Nearest is GRIFFIN (Dong et al., 2024). LoRA-Drop (Rajabzadeh et al., 2026) and RecycleGPT (Jiang et al., 2023) reuse states at trained or fixed schedules, a masked-diffusion family runs from dLLM-Cache (Liu et al., 2026) to SparseD (Wang et al., 2025), prompt-side caching follows Prompt Cache (Gim et al., 2024), CacheBlend (Yao et al., 2024), and Pensieve (Yu et al., 2023), and Future Lens (Pal et al., 2023) grounds the cross-token intuition. Speculative decoding (Leviathan et al., 2023, Medusa Cai et al., 2024a, EAGLE Li et al., 2024b) saves steps rather than bytes (composability: future work), and production engines (vLLM, TensorRT-LLM, SGLang, QServe) do not implement activation-sparse row skipping. Sparse attention and KV compression (StreamingLLM Xiao et al., 2024, H2O (Zhang et al., 2023), DuoAttention (Xiao et al., 2025), SnapKV (Li et al., 2024a), PyramidKV (Cai et al., 2024b), NSA (Yuan et al., 2025), SpargeAttn (Zhang et al., 2025), LServe (Yang et al., 2025a), and others) address attention segments, complementary to our FFN-side work.

5.4 Boundaries

(i) Against predictor-based skipping: we pay the gate tax and get zero false negatives, testing the extreme directly (whole-layer FFN skipping, 15.5% flip, Sec. 4.2). By injection, forcing reuse at false-negative rates of 0/1/5/10% moves long-chain accuracy $0.170 \rightarrow 0.150 \rightarrow 0.140 \rightarrow 0.100$ (Fig. 5, App. C), and with a real predictor granted its most favorable signal the mechanism drops 1.18% of deep-band live rows and 3.57% online, the 36.8% gate tax buying the absence of that divergence (App. C). (ii) Against engines: throughput-oriented serving is a different problem, left to future work (Sec. 8). (iii) Against TEAL: same kernel family, different operating point and fidelity standard, measured head-to-head at our rate on identical inputs (Sec. 4.3), their end-to-end results a different caliber (App. E).

6 Discussion

The ledger closed on itself. The measurement located the trap (the fixed term, with the proportional side at 98.2% of sustained bandwidth), the model predicted which escapes existed, and every

component of the stack entered by passing a prediction: the kernel because P1 says the row scan is escapable, int8 because the rotation-exclusion analysis dictates the recipe (Sec. 3.1), the 12-layer block because P3 prices finer gates below zero (0.8045×). The same predictions excluded predictor-based skipping (71% of generated tokens diverge granted its most favorable signal, App. C), whole-layer skipping (15.5% paired flip, Sec. 4.2), and per-layer gating (0.8045×, Sec. 4.5).

6.1 Redundancy migrates: from the spatial axis to the temporal one

Dense models' redundancy is primarily spatial, and sparse models have had it trained away (6.79% live-row rate), which is why it reappears on the temporal axis (the hypothesis of Sec. 3.3, measured in Sec. 4.4). There a training-free, zero-parameter mechanism (Sec. 5.3) runs at zero measured cost on natural text and 1.230× under sustained reuse (menu and drift cap: Sec. 4.4). The reused deep band is itself 99.8% dead at 0.2% live rows (predictor probe, App. C): the temporal axis exposes the same structure, not a separate source. Reuse buys time only in a steady regime, which is why the gate opens on self-generated repetition, not input entropy (Sec. 4.4).

Hardware scaling fortifies the argument. F is rate-independent while $a \cdot B/\beta$ shrinks as bandwidth grows, so on future hardware F occupies a growing fraction of total time. What strengthens is not the headline multiple but the importance of driving F toward zero (a prediction, cf. Limitation (i)).

6.2 Where the fixed-cost lens may extend

The rule is verified on row and layer skipping, and a minimal instance extends it to MoE routing (OLMoE-1B-7B decode latency is affine in the expert count, a ≈91.5 ms/tok k-independent frame against ≈1.2 ms per expert, App. B.14). Whether early exit and speculation pay the same fixed cost remains untested.

Future work: a zero-fixed-cost subgraph below block granularity, S12 glue fusion (≈0.431 ms avoidable), batch regimes (where the fixed-cost share shrinks and the relative win narrows), and cross-model and cross-hardware replication of the temporal regime.

7 Conclusion

Activation sparsity's missing speedup was never missing because of sparsity: it was absorbed by fixed costs that scale with nothing—a row scan inside the official kernel (escaping it buys 1.12–1.17×, Sec. 4.3), a wrong assumption in dense quantization recipes, and a decision tax in finer skipping (0.8045×). With those costs driven toward zero, we characterize a temporal stability regime absent in dense controls (0.999 cosine on 46.4% of adjacent-step pairs at 7B, 82.1% at 13B), exploited training-free at greedy agreement 1.0 (1.000× natural text, 1.230× only under sustained reuse). The composed stack reaches 1.38× and 1.55× over the official CUDA graph baseline and TEAL's graph kernel (2.66× over the official eager release form, cross-caliber), sits 1.36× above the bf16-dense ledger floor, and passes the hardest long-chain test we know (2.0% vs. 11.0% and 15.5%). Localizing the fixed-cost trap is the paper's new point. **For the mechanisms we verified, skip only at a granularity coarse enough that the decision costs ≈nothing, a rule we expect but do not demonstrate to extend to early-exit and speculative decoding.**

8 Limitations

(i) One family (ProSparse-LLaMA-2 7B/13B), one GPU, one scenario (RTX 5090, batch=1): transfer beyond the family is characterized but not universal, cross-generation hardware and batch transfer remain unverified, and prefixed-input transfer is measured rather than assumed (Sec. 4.4, App. F.6). (ii) Temporal reuse measures at zero cost on natural text (1.000×) and 1.230× under sustained reuse, its value lying in the phenomenon plus a bounded exploitability (≈2.1 ms of the 11.2 ms step). (iii) Comparisons stay within the sparsity track, and the dense baseline's low accuracy (16%) leaves discrimination to paired statistics. (iv) The fidelity protocol is our own design (pre-registered, noise-floor calibrated).

REPRODUCIBILITY STATEMENT

(i) Measurement disciplines: caliber unity (eager vs. graph differ 2.4–3.0×, no cross-caliber arithmetic). Per-rep deduplicated footprint $\geq 8\times$ L2 (768 MB) for segment/kernel bandwidth. Correctness before speed (every config replays against dense, rel_err < 0.03). Teacher forcing, plus the released three-gate self-check (kernel against the dense FFN, graph replay against the engine's eager forward, and the reuse path against full-compute). Bit-width monotonicity and no σ-clipping in quantization. Bandwidth reported on useful bytes (write amplification included). Fixed event-timing argument order and identical per-arm inputs. Fixed-vs-proportional decomposition before explaining any "low efficiency". (ii) Caliber trio: bandwidth denominator 1,689.8 GB/s (sole). Invalidation threshold 1,705.1 GB/s. Eff ceiling 1.0090. (iii) Stack labels: B = 8.066 ms graph baseline (artifact: exactly 128+33 GEMV kernels). E = engine/eager. Cross-stack comparisons are marked non-comparable. (iv) Zero-trust verification: all paper numbers independently recomputed from raw records (Sec. 4.2 gives one caliber-trap example). (v) Artifacts: experiment scripts, data JSONs (locally archived and on the measurement host, dual-ended with hashes), and full revision history. Code—including the row-skipping kernel, the dual-graph reference implementation, and the fidelity-protocol scripts—will be released with the paper (anonymized for review). The integrated harness that produced the waterfall levels currently runs on a host that is offline: its archived result records ship with the artifact, the packaged components reproduce its component math, and App. G records the portability rules for re-verifying those levels on a new machine.

Reproducibility contract: the graph-caliber ratio ladder (8.042 / 7.019 / 6.291 ms, with the L3 denominators 8.707 / 9.765 / 12.186 ms behind the 1.38× / 1.55× / 1.94× claims, and the 1,689.8 GB/s bandwidth anchor) travels between machines within 1.5%, while the absolute eager-class figures (16.752 / 12.657 ms) and the same-session dense comparators (12.186 ms) are not meant for cross-machine matching, so reproduce the ratios rather than the absolutes.

ETHICS STATEMENT

This work studies inference efficiency of an existing open model and introduces no new capability. Societal impact does not exceed that of baseline LLM inference. Faster inference reduces energy per token, a net positive on the margin.

GENERATIVE AI USE STATEMENT

In this work, we used large language models for coding assistance (implementation support and debugging of experiment code), literature search and summarization, and drafting, editing, and polishing of the manuscript under the authors' direction. All experimental code was executed and verified by the authors, and every reported number was independently recomputed by the authors from raw experimental records (per-question logs, per-layer profiles, per-rep timings). We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

Yash Akhauri, Ahmed F. AbouElhamayed, Jordan Dotzel, Zhiru Zhang, Alexander M. Rush, Safeen Huda, and Mohamed S. Abdelfattah. Shadowllm: Predictor-based contextual sparsity for large language models. In *EMNLP 2024*, 2024. arXiv:2406.16635.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L. Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. In *NeurIPS 2024*, 2024. arXiv:2404.00456.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *ICML 2024*, 2024a. arXiv:2401.10774.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling, 2024b. arXiv:2406.02069 (preprint).

Jaewon Cheon and Pilsung Kang. Countdown: Contextually sparse activation filtering out unnecessary weights in down projection. In *EMNLP 2025*, 2025. arXiv:2505.17701.

Harry Dong, Beidi Chen, and Yuejie Chi. Prompt-prompted adaptive structured pruning for efficient LLM generation. In *First Conference on Language Modeling (COLM)*, 2024. arXiv:2404.01365.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. In *ICLR 2023*, 2023. arXiv:2210.17323.

In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. In *MLSys 2024*, 2024. arXiv:2311.04934.

Ajay Jaiswal, Bodun Hu, Lu Yin, Yeonju Ro, Shiwei Liu, Tianlong Chen, and Aditya Akella. Ffn-skipllm: A hidden gem for autoregressive decoding with adaptive feed forward skipping. In *EMNLP 2024*, 2024. arXiv:2404.03865.

Yuchu Jiang, Yue Cai, Xiangzhong Luo, Jiale Fu, Jiarui Wang, Chonghan Liu, and Xu Yang. d^2cache: Accelerating diffusion-based llms via dual adaptive caching. In *ICLR 2026*, 2026. arXiv:2509.23094.

Yufan Jiang, Qiaozhi He, Xiaomin Zhuang, Zhihua Wu, Kunpeng Wang, Wenlai Zhao, and Guangwen Yang. Recyclegpt: An autoregressive language model with recyclable module, 2023. arXiv:2308.03421 (Technical Report).

Filipe Laitenberger, Dawid Kopiczko, Cees G. M. Snoek, and Yuki M. Asano. What layers when: Learning to skip compute in llms with residual gates. In *ICLR 2026*, 2026. arXiv:2510.13876 (method name: GateSkip).

Donghyun Lee, Je-Yong Lee, Genghan Zhang, Mo Tiwari, and Azalia Mirhoseini. Cats: Contextually-aware thresholding for sparsity in large language models. In *COLM 2024*, 2024. arXiv:2404.08763.

Younjoo Lee, Seungkyun Dan, Junghoo Lee, Jaiyoung Park, and Jung Ho Ahn. Dyllm: Efficient diffusion llm inference via saliency-based token selection and partial attention. In *ICML 2026*, 2026. arXiv:2603.08026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML 2023*, 2023. arXiv:2211.17192.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In *NeurIPS 2024*, 2024a. arXiv:2404.14469.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. In *ICML 2024*, 2024b. arXiv:2401.15077.

Yesheng Liang, Haisheng Chen, Zihan Zhang, Song Han, and Zhijian Liu. Paroquant: Pairwise rotation quantization for efficient reasoning llm inference. In *ICLR 2026*, 2026. arXiv:2511.10645.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. In *MLSys 2024*, 2024. arXiv:2306.00978.

James Liu, Pragaash Ponnusamy, Tianle Cai, Han Guo, Yoon Kim, and Ben Athiwaratkun. Training-free activation sparsity in large language models. In *ICLR 2025*, 2025a. arXiv:2408.14690.

Kai Liu, Bowen Xu, Shaoyu Wu, Xin Chen, Hao Zhou, Yongliang Tao, and Lulu Hu. La rosa: Enhancing llm efficiency via layerwise rotated sparse activation. In *ICML 2025*, 2025b. arXiv:2507.01299.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant: Llm quantization with learned rotations. In *ICLR 2025*, 2025c. arXiv:2405.16406.

Zhiyuan Liu, Yicun Yang, Yaojie Zhang, Junjie Chen, Chang Zou, Qingyan Wei, Shaobo Wang, Yichen Zhu, and Linfeng Zhang. dllm-cache: Accelerating diffusion large language models with adaptive caching. In *ICML 2026*, 2026. arXiv:2506.06295.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. Deja vu: Contextual sparsity for efficient llms at inference time. In *ICML 2023*, 2023. arXiv:2310.17157.

Yuqi Luo, Chenyang Song, Xu Han, Yingfa Chen, Chaojun Xiao, Xiaojun Meng, Liqun Deng, Jiansheng Wei, Zhiyuan Liu, and Maosong Sun. Sparsing law: Towards large language models with greater activation sparsity. In *ICML 2025*, 2025. arXiv:2411.02335, PMLR 267.

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models, 2025. arXiv:2505.15781 (preprint).

Iman Mirzadeh, Keivan Alizadeh, Sachin Mehta, Carlo C. Del Mundo, Oncel Tuzel, Golnoosh Samei, Mohammad Rastegari, and Mehrdad Farajtabar. Relu strikes back: Exploiting activation sparsity in large language models, 2023. arXiv:2310.04564 (preprint).

Quan Nguyen-Tri, Mukul Ranjan, and Zhiqiang Shen. Attention is all you need for kv cache in diffusion llms, 2025. arXiv:2510.14973 (preprint, method name: Elastic-Cache).

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron C. Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In *CoNLL 2023*, 2023. arXiv:2311.04897.

Hossein Rajabzadeh, Maryam Dialameh, Chul B. Park, Il-Min Kim, and Hyock Ju Kwon. Lora-drop: Temporal lora decoding for efficient llm inference, 2026. arXiv:2601.02569 (preprint).

Jiho Shin, Hoeseok Yang, and Youngmin Yi. Grasp: Group-based prediction of activation sparsity for fast llm inference. In *DAC 2025*, 2025a. DOI 10.1109/DAC63849.2025.11132899 (title corrected to the published form, Round 10 audit).

Jiho Shin, Hoeseok Yang, and Youngmin Yi. Sparseinfer: Training-free prediction of activation sparsity for fast llm inference. In *DATE 2025*, 2025b. arXiv:2411.12692.

Chenyang Song, Xu Han, Zhengyan Zhang, Shengding Hu, Xiyu Shi, Kuai Li, Chen Chen, Zhiyuan Liu, Guangli Li, Tao Yang, and Maosong Sun. Prosparse: Introducing and enhancing intrinsic activation sparsity within large language models, 2024a. arXiv:2402.13516 (preprint).

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu. In *Proceedings of the 30th Symposium on Operating Systems Principles (SOSP '24)*, pp. 590–606, 2024b. doi: 10.1145/3694715.3695964. arXiv:2312.12456.

Yixin Song, Haotong Xie, Zhengyan Zhang, Bo Wen, Li Ma, Zeyu Mi, and Haibo Chen. Turbo sparse: Achieving llm sota performance with minimal activated parameters, 2024c. arXiv:2406.05955 (preprint).

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. In *ICML 2024*, 2024. arXiv:2402.04396.

Zeqing Wang, Gongfan Fang, Xinyin Ma, Xingyi Yang, and Xinchao Wang. Sparsed: Sparse attention for diffusion language models, 2025. arXiv:2509.24014 (preprint).

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding, 2025. arXiv:2505.22618 (preprint).

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *ICML 2023*, 2023. arXiv:2211.10438.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR 2024*, 2024. arXiv:2309.17453.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *ICLR 2025*, 2025. arXiv:2410.10819.

Shang Yang, Junxian Guo, Haotian Tang, Qinghao Hu, Guangxuan Xiao, Jiaming Tang, Yujun Lin, Zhijian Liu, Yao Lu, and Song Han. Lserve: Efficient long-sequence llm serving with unified sparse attention. In *MLSys 2025*, 2025a. arXiv:2502.14866.

Yiheng Yang, Yujie Wang, Chi Ma, Lei Yu, Emmanuele Chersoni, and Chu-Ren Huang. Sparse brains are also adaptive brains: Cognitive-load-aware dynamic activation for llms, 2025b. arXiv:2502.19078 (preprint).

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving for rag with cached knowledge fusion, 2024. arXiv:2405.16444 (preprint).

Lingfan Yu, Jinkun Lin, and Jinyang Li. Stateful large language model serving with pensieve, 2023. arXiv:2312.05516.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *ACL 2025*, 2025. arXiv:2502.11089.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. In *ICML 2025*, 2025. arXiv:2502.18137.

Zhengyan Zhang, Yixin Song, Guanghui Yu, Xu Han, Yankai Lin, Chaojun Xiao, Chenyang Song, Zhiyuan Liu, Zeyu Mi, and Maosong Sun. Relu^2 wins: Discovering efficient activation functions for sparse llms, 2024. arXiv:2402.03804 (preprint).

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *NeurIPS 2023*, 2023. arXiv:2306.14048.

Yang Zhou, Zhuoming Chen, Zhaozhuo Xu, Victoria Lin, and Beidi Chen. Sirius: Contextual sparsity with correction for efficient llms, 2024. arXiv:2409.03856 (preprint).

A Full ablation tables

This appendix expands the ablation summaries of Sec. 5 into the component-level tables that the main text cites. All entries are at the measurement calibers of the corresponding main-text section.

A.1 Component decomposition (the end-to-end waterfall)

The waterfall table is in Sec. 4.3. The same-session re-measurements follow.

Source: paper Sec. 4.3 (Fig. 6) and `audit_results/night_0915/iclr27_waterfall_repro.json` *(A-graph four arms) and* `s7c_clean_e2e.json` *(stock 16.752 / stock_where 12.657 / engine_eager 14.229 / engine_graph 11.728).*

The same session re-measures the sparse arm of the identical stack at **8.150 ms** (1.3% reproduction of the 8.042 anchor) and the engine-dense graph stack at **12.186 ms** (primary matched-caliber dense denominator, Sec. 4.3). The dense/sparse isolation measured inside that session is **1.4952** (the two arms differ only in the up/down row mask), while the same dense arm against the 8.042 anchor gives the cross-session figure 1.5153.

A.2 Kernel configuration sweep (head-to-head and matched-rate cross-check)

The fixed-cost flat-line curve of Fig. 3 is the real-weight replay (three measured bands). The matched-rate cross-check between synthetic and real-distribution runs gives 0.990 (shallow) and 0.999 (mid) (`c032_replay_20260916.json`). *Source: paper Sec. 2.2 / Sec. 3.1 and* `audit_results/` *mirror of* `c032_live.json` *(server) and* `c032_replay_20260916.json` *(server).*

Opponent	Ours / theirs (at deployment live-row rate 6.79%)	Caliber
Official ProSparse row-skipping kernel	**1.12–1.17**×	graph, identical inputs
TEAL (2025)	**1.27–1.97**×	graph, identical inputs

Ingredient	Chosen setting	Rationale / measured evidence
Weight bit-width	int8 (W8A16)	bit-width monotonicity, int4 fails (KL 0.319), int8 KL 0.00863 < 0.05
Activation bit-width	bf16 (A16)	activations carry the live-row heavy tail, quantizing them loses the salient live outputs
Rotation	**none**	Hadamard-128 rotation destroys dead-row exact zeros: skipping the same positions after rotation incurs 34.4–48.1% output relative error (native-basis skip = 0.0)
Zero-point	integer, asymmetric	full range preserved, no outlier clipping
Granularity	g128 (1 scale/zero per 128 elements)	per-tensor and per-row scales are biased by dead-row zeros, g128 is the measured knee
σ-clipping	none	clipping at 3σ costs the sparse model a median 0.841 output-relative error vs. 0.531 for the dense control

Fairness cross-check. In the official repository's own test entry the official kernel beats its own dense path by **1.20–1.45**× (≈1.5× its README), confirming the baseline is competitive. A config-fixed TEAL rerun (c035c, repairs a sparsity-binning defect) gives **1.29–1.95**×, within 1.4% of the quoted 1.27–1.97×. The endpoints map to TEAL's home operating point (act=0.50, 1.29×) and our deployment point (act=0.07, 1.95×). The official/ours cross-over sits at ≈8–10% live rows. The deployment mean (6.79%) is below it, where our kernel wins. The shallow high-live-row band (12–13%) is the only region where the reference wins, and it is outside the deployment regime.

A.3 Quantization recipe matrix

The recipe is fixed by the sparsity×quantization sensitivity rule (Sec. 3.1). *Source: paper Sec. 3.4, Sec. 4.2, release* `quant/rtn.py` *(mechanism check).*

A.4 int4 arms detail (V2 RTN, V4 AWQ)

Same five-projection quantization scope and the same 200-question GSM8K-4-shot protocol as the int8 arm. *Source: paper Sec. 4.2 (Fig. 2).*

Both int4 arms exceed the 10% break line. They are statistically indistinguishable from each other (RTN Wilson CI [7.4%, 16.1%], AWQ [9.4%, 18.9%]) and 6.8× the int8 rate. AWQ raises accuracy and chain completeness over RTN but also raises the gating flip rate, so on the metric that gates the claim int4 stays above the 10% line. † V4's selfcheck is the same q0-reference first-token logit deviation as V1–V3.

A.5 RTN and AWQ protocols, including calibration set

- **RTN (V2).** Standard round-to-nearest, per-group (g128) asymmetric integer quantization, **no calibration set** (RTN requires no data). Identical five-projection scope and 200-question protocol as all other arms.
- **AWQ (V4).** The standard activation-aware pipeline exactly as shipped by the released AWQ implementation: w4, group size 128, integer zero-point, activation clipping and duo-scaling enabled. **Calibration set: 128 samples × 512 tokens from wikitext** (`gsm8k_-4shots.json` is the evaluation prompt, the calibration corpus is the standard AWQ wikitext recipe). Scope: RTN and AWQ are the only two recipes benchmarked under this identical protocol. **GPTQ and QServe are not benchmarked** (scope of this appendix). The safe-line claim (int4 outside tolerance on sparse models) is therefore scoped to RTN + AWQ under this protocol.

Arm	Recipe	acc	Chain	Flip vs. dense	q0-ref KL	Selfchk
V2 int4	RTN (round-to-nearest, no calib.)	0.070	0.355	**0.110 (20 bad / 2 good)**	0.0671	0.0671
V4 int4	AWQ (w4, g128, zero-point, clip + duo-scale, official impl.)	0.115	0.440	**0.135 (18 bad / 9 good)**	0.0701†	0.0701

Denominator	ms/token	Caliber	Speedup of our gate8 (6.291)
Official release form (stock HF, eager)	16.752	eager (download)	2.66× (cross-caliber)
Where-fixed dense (torch.where, eager)	12.657	eager	2.01× (cross-caliber)
Engine dense + whole-step graph	12.186	**graph (primary dense)**	**1.94**×
Official kernel graph (captured in our harness)	8.707	graph	1.38×
TEAL graph (this model)	9.765	graph	1.55×
Our engine bf16 (L1)	8.042	graph	1.08× (vs. official graph)
Our eager engine (aside)	≈11.26 (= 0.89× the where-fixed dense)	eager	—

A.6 Same-scope module list

All quantization arms (RTN, AWQ, int8) quantize the **identical per-layer projection set** — the gate/up/down FFN projections plus the q/k/v/o attention projections (the "five-projection scope" of Sec. 4.2). Differences between arms therefore isolate the *recipe*, not the module coverage. The gate projection itself is the FATReLU ground-truth gate and is held at bf16 in every fidelity arm (it must remain exact for the zero-false-negative contract), while the L3 speed rung quantizes it separately. *Source: paper Sec. 4.2, Sec. 3.4.*

A.7 Refusal to rotate (rationale)

Rotations that help dense models (QuIP#, SpinQuant, ParoQuant) are harmful on activation-sparse models for one structural reason: they remix the dead rows' *exact* zeros with live outliers, destroying the exact-zero structure the zero-false-negative contract depends on. Two direct measurements on ProSparse-LLaMA-2-7B (layers 16/24/28, 4,096 tokens from the GSM8K 4-shot prompts, dense Mistral-7B layers 16/24 as control, dead-row fractions 0.88–0.97):

- Skipping native-basis dead rows is mathematically exact: output relative error **0.0**.
- After a Hadamard-128 block rotation of the same operator, the dead positions fill to ordinary-coordinate level and skipping the same positions incurs **34.4–48.1%** output relative error.
- σ-clipping at 3σ costs the sparse model a median **0.841** output-relative error vs. **0.531** for the dense control.

The released int8 recipe accordingly uses **no rotation** and **no σ-clipping**. *Source: paper Sec. 3.1 mechanism check and Sec. 5.2.*

A.8 Baseline ladder (closed at the 12.186 / 8.042 rungs in hand)

All denominators are labeled with their caliber. The matched-graph-caliber ladder is the portable class (Sec. 4.1, App. G). *Source: paper Sec. 4.1 / Sec. 4.3,* `audit_results/official_kernel_e2e_NEW.json`, `teal_e2e_NEW.json`, `s7c_clean_e2e.json`.

Cross-session retest on a second host reproduces the graph-caliber anchors within <1% (official graph 8.712 vs. 8.707, TEAL 9.672 vs. 9.765, sustained read 1,684.3 vs. 1,689.8 GB/s) while the eager-stack absolutes move by up to 17.7% (19.712 vs. 16.752 ms). The graph-caliber ladder is

Arm	acc	Chain completeness	Paired flip rate vs. dense	Selfcheck strength
V0 dense (bf16 baseline)	0.160	0.510	–	–
V1 int8 (in-stack)	0.140	**0.510 (implementation replay bitwise-identical)**	**0.020 (4 bad / 0 good)**	0.0096
V2 int4	0.070	0.355	**0.110 (20/2)**	0.0671
V4 int4 (AWQ, well-tuned)	0.115	0.440	**0.135 (18/9)**	0.0701†
V3a head pruning	0.130	0.500	0.040 (7/1)	0.0189
V3b whole-layer FFN skip	0.075	**0.190 (collapse)**	**0.155 (24/7)**	0.1595

therefore the only class used for cross-machine or cross-session claims. *Source: paper Sec. 4.1 and App. G (G.3).*

A.9 Concrete execution: kernel, lagged-cosine, int8, composition

The lean row-skipping kernel (S7+S8: 1.236 ms, 5.08% of bytes). Three cuts. **Detection hoisted:** liveness comes from the ground-truth gate (zero false negatives), one gate value per row instead of a visit to the weight row. **Masked reads:** the live predicate applies to the weight loads themselves, so dead rows contribute no bytes and reads stay block-aligned rather than becoming a row-wise gather, a measured choice (a pure row-gather probe manages only 904.9 GB/s at 1% density and 1,596 GB/s at 14%, while the up/down segment reaches a marginal 1,659 GB/s at 6.79% live, 98.2% of sustained, Sec. 3.1). **Fixed-count reduction:** the down projection combines a fixed number of times. **Correctness gate:** every configuration replays bitwise or within-tolerance against the dense path (rel_err < 0.03) before timing.

The lagged-cosine dual-graph mechanism (deep FFN). Decisions must cost no synchronization and no extra kernels. Each step writes the 12 deep layers' (L20–31) post-LN2 hidden states into a static buffer and computes cosine(x_{t−1}, x_{t−2}) in-graph, with the host reading 12 floats back asynchronously and opening for step t only if all layers exceed 0.9995. Graph A computes everything and refreshes the true deep-FFN outputs and graph B skips the deep FFN and adds the cached outputs to the residual stream, each step replaying one graph chosen from the *previous* step's signal, so a gate failure returns to A and reuse always draws on the most recent *true* values.

The sparsity-aware int8 recipe (S1: 2.962 ms, 54.8% of bytes). W8A16, **no rotation**, **integer zero-point asymmetric** (full range preserved), **g128 granularity** (per-tensor scales are biased by dead-row zeros on sparse distributions), served by a custom W8A16 GEMV and claimed only as a component. **Composition:** whole-step CUDA graph (cuts S12 launch tax) → S1 int8 (halves the largest segment) → lean row-skip (fixes S7+S8's fixed term) → temporal reuse (deep FFN unread on reuse steps). The four components are multiplicative, and the stack resists trivial re-assembly: four requirements, each independently measured. Without ground-truth dead rows, skipping either breaks fidelity (the FFN-skip arm's 15.5% flip rate) or pays predictor compensation. Without the right recipe, sparse-distribution quantization collapses (Sec. 3.1). Without the zero-false-negative contract, the fidelity protocol fails (Sec. 3.4). Without the temporal mechanism, the deep-FFN reuse bytes are simply invisible (a dense model has ≈none, Sec. 3.3). **None of the four pieces is new in isolation.** What the accounting frame supplied is the order and the reason: which fixed term to cut first, why the recipe must not rotate, and why reuse needs a ground-truth gate, and a stack assembled by profiling alone would have to rediscover each of those constraints by trial.

Six-arm fidelity table. Rows bold the proposed arm, not the row-optimal value, and † marks the same logit deviation as V1–V3.

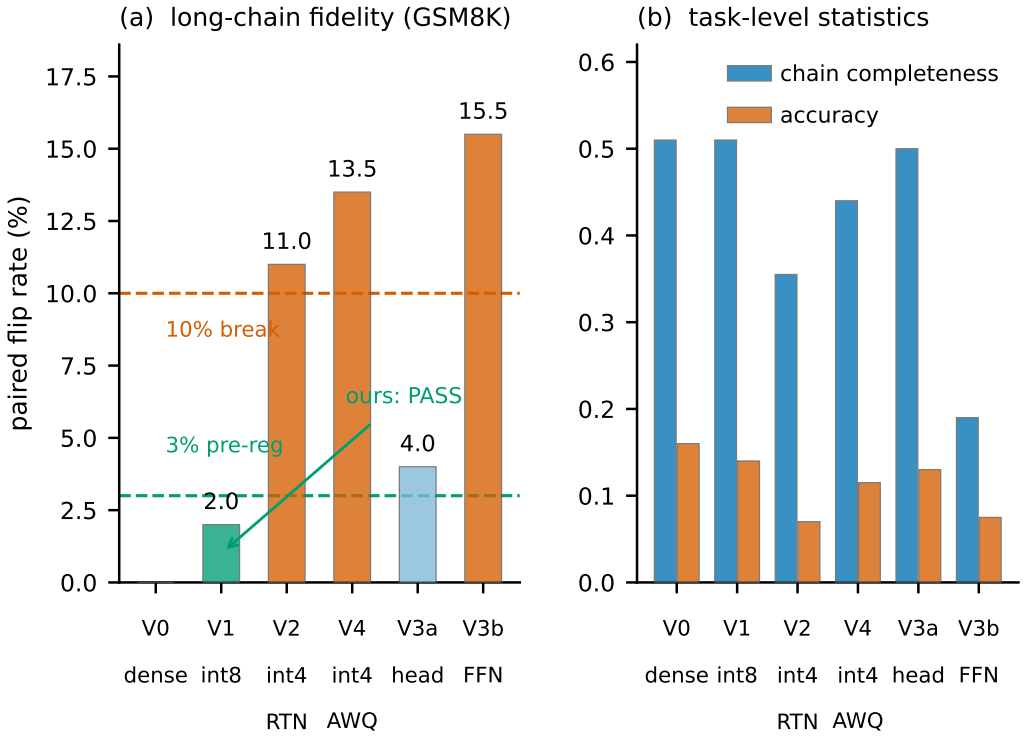

Figure 2: Two panels. Left: paired flip rate (%) for the six arms labeled V0 dense (reference), V1 int8 (ours), V2 int4 RTN, V4 int4 AWQ, V3a head pruning, V3b whole-layer FFN skip, with pre-registered 3% / 10% threshold lines, V1 marked "ours: PASS", both int4 arms above the 10% line. Right: chain completeness and accuracy (y = rate, 0–1), grouped bars.

Item	Value
GPU	NVIDIA GeForce RTX 5090 (sm_120, 96 MB L2)
CPU / memory / OS	Xeon 8470Q ×2 / 754 GB / Ubuntu 22.04
Software	torch 2.8.0+cu128, triton 3.4.0
Measured sustained bandwidth (caliber anchor)	**1,689.8 GB/s** (large-buffer sequential read)
Invalidation threshold	1,705.1 GB/s (above this ⇒ L2-resident working set, discarded)
Model	ProSparse-LLaMA-2-7B (official SparseLLM weights, bf16, FATReLU threshold 0.01)
Deployment live-row rate	6.79% (measured)
Also used	ProSparse-LLaMA-2-13B (Sec. 4.4)

B Experimental Details

B.1 Environment

Source: paper Sec. 4.1, `audit_results/hw_fingerprint_full_20260920.json`*,* `audit_results/fp_stability.json`*.*

Recomputation check (no GPU needed). The ledger is verifiable from public facts alone. The architecture's public dimensions (d = 4096, FFN = 11,008, 32 layers, bf16) with the 6.79% deployment live-row rate compute a per-step weight traffic of 7.83 GB against the 7.84 GB measured, with segment shares qkvo 54.8%, gate 36.8%, and up+down 5.0% matching Fig. 1 to within 0.1 percentage points. The bandwidth anchor (1,689.8 GB/s) is 94.3% of the card's 1,792 GB/s specification. A reader with the data sheet can verify the ledger without running anything.

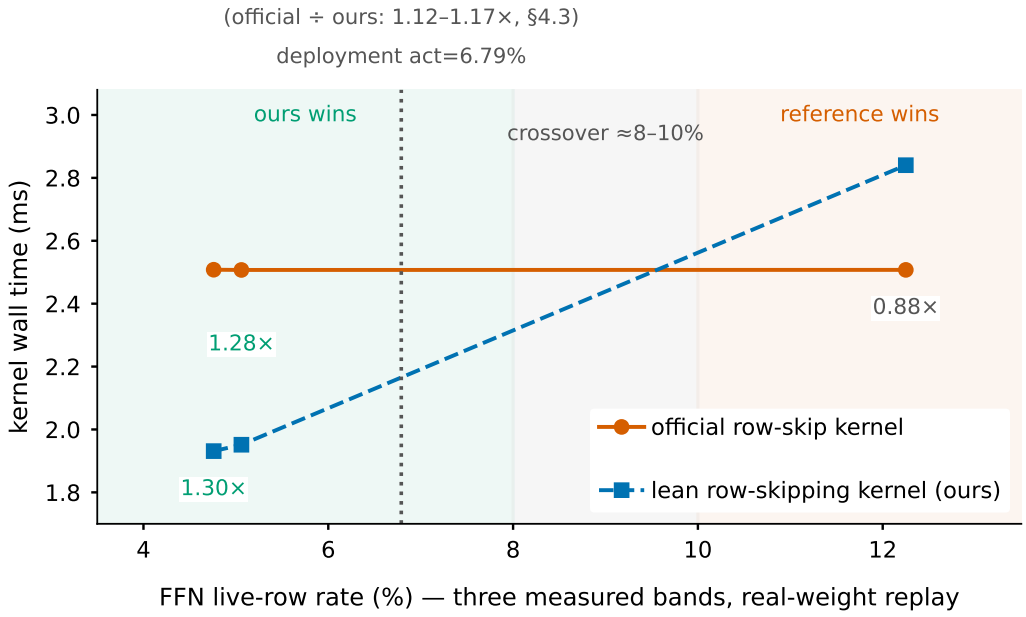

Figure 3: Time-vs-live-row-rate curves on the real-weight replay, three measured bands. x = FFN live-row rate (%), y = kernel wall time (ms). The official curve stays nearly flat as the rate falls (the fixed-cost trap made visible) while ours keeps decreasing, crossing over at ≈8–10% and winning at the deployment point (6.79%), with the measured head-to-head ratios annotated. The matched-rate cross-check is reported in App. A.2.

B.2 Prompts

- **Long-chain fidelity (Sec. 4.2, App. A/B/C).** GSM8K 4-shot chain-of-thought, ×200 questions, paired per question. Larger reruns: n=1,000 and n=1,319 (`e1_cot_arm_V1_-n319ext.json`), the latter extending the 1,000-question pool by a further 319 questions.
- **Temporal-reuse profiling (Sec. 4.4).** 200 wikitext segments, 19,000 adjacent-step pairs per layer, teacher-forced continuation (8 segments × 32 continuation positions for the fired-step distribution, 768 teacher-forced positions per model for the bitwise probe).
- **Prefix cross-check (Sec. 4.4).** 24 segments spanning two corpora (parquet / txt).
- **Reuse loops.** 256-step and 512-step self-generated repetition loops (two repetition regimes) for drift and long-horizon checks.

B.3 Pre-registered criteria (full text)

The fidelity protocol (Sec. 3.4) was fixed before measurement. Its implementation layer is bitwise or within-tolerance replay plus a released self-check, and its task layer is paired flip rate with chain completeness. Verbatim criteria:

1. **Distribution.** Top-K truncated KL, reported as percentiles p50/p90 (never the mean). Passing threshold: **KL < 0.05**. Where a selector decides reuse, percentiles are reported over all steps *and* separately over the steps the selector fired on.
2. **Decision.** argmax (token-identity) agreement = 1.0 under greedy decoding.
3. **Task (paired flip).** On GSM8K 4-shot CoT, paired per question: **flip rate < 3% passes. Flip rate > 10% breaks.** The 3% / 10% lines are the two pre-registered decision boundaries.
4. **Chain completeness** is reported alongside (no fixed pass/fail line, it is a secondary discriminator rather than a gate).

Eval set	ΔPPL	ΔNLL (95% CI)	KL [p50, p90]	top-1	Verdict
8k	**+3.39%** (fails <1%)	**+0.0333**	[0.0023, 0.0208]	0.942	fails gate 8
32k	**−1.25%**	**−0.0126**	[0.0023, 0.0192]	0.950	within <1%
133k	**−1.38%**	**−0.0139**	[0.0026, 0.0185]	0.948	revised gate 0.030 passes, task flip 2.0% passes

n	flip rate	bad / good (or total flips)	Wilson 95% CI
200	2.0%	4 bad / 0 good (4/200)	[0.78%, 5.03%]
1,000	2.60%	—	[1.78%, 3.78%]
1,319	**2.81%**	17 bad / 20 good (37/1,319)	[2.04%, 3.84%]

5. **Auxiliary gates.** Teacher forcing is used instead of autoregressive comparison (an early mistake collapses autoregressive comparisons and is unattributable). Bit-width monotonicity is required for any quantization claim (if a higher bit-width is not lossless, the measurement is broken rather than merely weaker).
6. **Zero-false-negative contract.** The kernel's skipped set must equal the gate's ground-truth set, with no false positives or false negatives.
7. **ΔNLL gate (quantization).** Initially pre-registered at 0.00995 with a CI upper bound below it. On the 133k set the CI upper is 0.018, so the gate was revised to **0.030** together with the multi-metric definition (ΔNLL CI + KL + task flip) used throughout.
8. **ΔPPL gate (quantization).** Pre-registered at **< 1%** degradation.

Source: paper Sec. 3.4, Sec. 4.2.

B.4 INT8 CALIBER-SENSITIVITY MATRIX (FULFILLED FROM ARCHIVED RECORD)

The matrix varies the *evaluation token-set size* (caliber) of the int8 recipe, over nested subsets of the wikitext holdout (8k ⊂ 32k ⊂ 133k). Archived truth values (the full per-cell matrix is now in the local archive). These are the nested-subset matrix protocol (8k ⊂ 32k ⊂ 133k, sl=1024, k_top=64). The small-set PPL (+6.24%) and the 133k ΔNLL (−0.00405, CI crossing 0) quoted in the main text come from the s117 / s121 protocols and are reported separately, never merged with this matrix:

Source: `int8_caliber_matrix_20260916.json` *(archived locally, s119/cot_fid-V1 quantization math, nested 8k ⊂ 32k ⊂ 133k subsets of the wikitext holdout, top-K truncated KL, sl = 1024, k_top = 64).* The gate numbers that Sec. 4.2 quotes come from independent sessions s117–s122. They are shown side by side here and not merged into a single pooled estimate. The disagreement concentrates at the smallest set, and the 32k and 133k rows agree in sign. The Sec. 4.2 verdict still rests on the large-set triple (ΔNLL CI + KL + task flip) rather than on any single small-set PPL.

B.5 INT8 POWER / CI ANALYSIS, INCLUDING LARGER-N BOUNDS

The paired flip rate is the gating task metric. Its Wilson 95% intervals at every disclosed sample size:

Interpretation (disclosed as-is, not post-selected): the point estimate passes the pre-registered 3% line at **every** disclosed sample size, but the interval's upper edge extends past 3% at both n=200 and n=1,319. The direction is near-symmetric (n=1,319: 17 bad vs. 20 good flips ⇒ borderline-question noise rather than directional degradation). Chain completeness (n=1,319): 0.491 (dense) vs. 0.480 (int8). Accuracy even slightly favors int8 (0.123 vs. 0.121).

Statistical power / noise floor. A noise-floor arm (V0G1, cross-run determinism) shows **0.0** flip rate, confirming the 2.0% is real signal, not measurement noise. *Source: paper Sec. 4.2,* `audit_results/e1_cot_arm_V1_n319ext.json`.

Threshold	Sparse deep band (L20–31)	Dense control (same-size, per layer)
≥ 0.99	62.7%	0 / 19,000 (0%)
≥ 0.995	58.9%	0 / 19,000
≥ 0.999	**46.4%**	0 / 19,000
≥ 0.9995	25.4%	0 / 19,000
≥ 0.9999	0.05%	0 / 19,000

Condition (ProSparse-LLaMA-2-7B, sparse)	mean cos	r999
parquet pool, no prefix	0.9589	**0.426**
txt pool, no prefix	0.9612	**0.323**
txt pool, one `" ."` prepended (non-BOS)	0.9651	**0.0001**
parquet pool, BOS prepended	0.8516	0
txt pool, BOS prepended	0.8581	0
dense Mistral (both conditions)	0.498 / 0.509	0

B.6 Stability threshold sweep (five thresholds, 200 segments, per-layer)

A five-threshold sweep over {0.99, 0.995, 0.999, 0.9995, 0.9999}, 200 wikitext segments, 19,000 adjacent-step pairs per layer, same protocol, sharpens the sparse/dense separation. *Source: paper Sec. 3.3 (Fig. 9) and* `iclr27_m_sweep_20260919.json` *(server).*

The dense control shows zero exceedances at every layer for every threshold. The sparse deep band decays monotonically, with the operating gate (0.9995) inside the profile. Rule of three bounds the dense-side rate at $\leq$0.016% (95%), $\approx$2,900$\times$ below the sparse deep-band rate at 0.999.

B.7 Anchor reproduction of the original full-scan value at 46.4%

The 46.4% deep-band rate was reproduced across reruns (0.4641 on one re-check, the parquet no-prefix pool gives 0.426, the txt no-prefix pool 0.323 — both are the same phenomenon under different corpora/protocols, see B.8). The 46.4% figure quoted in the main text is the L20–31 average under the 200-segment protocol. *Source: the two-pass reproduction record (archived) and paper Sec. 4.4.*

B.8 Cross-check per-cell statistics (prefix cross-check)

A 24-segment cross-check spanning two corpora measures the deep-band ($\geq$0.999) rate before/after prepending a single ordinary token (different sample pool from the 200-segment main protocol). *Source: paper Sec. 4.4 and* `temporal_redundancy_generalization_report_20260918.md` *(six-cell matrix).*

Reading: the precise tail (r999) exists only under no-prefix construction. Prepending any single token collapses it, while the mean cos rises (the collapse is a tail-structure disappearance, not a global degradation). BOS further depresses the mean (0.96$\rightarrow$0.86). Dense models are indifferent to all conditions. The main-text 46.4% uses the no-prefix deployment protocol.

B.9 Reuse fidelity frontier (gate threshold $\times$ chain cap $\times$ reuse-block size $\times$ model)

The redundancy space of Fig. 8 / App. F is the joint sweep of the four knobs. Here *deep ring* and *shallow ring* name the sweep's two repetition-profile inputs (the EOS and punctuation loops), not layer bands. Compact frontier (same-run measurements, all graph caliber): *Source:* `appendix_G_H_tables_v213.md` *(F.1/F.2) and* `audit_results/reuse_boundary_b2/b3*`, `tf_bitwise_probe_20260920.json`, `extra_skip.json`.

Reuse-block-size axis (7B shallow, τ=0.9995): 12-layer block 0.367 · 8-layer 0.938 · 6-layer 0.969 · 4-layer 0.984 (coverage, drift comparable). The 12-layer block is the depth-profile consequence, not a hyperparameter search. *Fired-step distribution:* on natural text at the default 7B gate, the gate opens on 2.34% of positions. At opened positions divergence is 0.19 nats (median) / 0.39 (max) — the mechanism's own error lives in the tail an all-step median cannot see, which is why fired steps are

Model · regime	τ	cap	coverage	agree	fired-step KL [p50, p90, max]	speed
7B · deep ring (default gate)	0.9995	—	0.984	1.0	[1.12, 1.65, 1.70]	**1.230**×
7B · deep ring	0.9999	8	0.789	1.0	[0.144, 0.191, 0.517]	1.180×
13B · shallow ring (default gate)	0.9995	—	0.984	1.0	[0.011, 0.036, 0.066]	~1.19×
13B · shallow ring	0.9999	8	0.805	1.0	[0.0003, 0.025, 0.027]	**1.154**×
13B · deep ring, m = 0.15	0.9999	8	0.672	**1.0**	[2.5e-4, 0.0995, 0.203]	—

reported separately (`tf_bitwise_probe_20260920.json`: 256 positions, 6 fired, kl_p50 0, kl_p90 0, kl_max 0.389, kl_fired_p50 0.189, kl_fired_p90 0.389). The same gate in the reuse-heavy regime (row 1) carries its fired steps deep in reuse chains, where the distribution runs [1.12, 1.65, 1.70] — same gate, different input regime, reported by construction and never merged. *Guarded setting:* the chain cap bounds accumulated drift (Sec. 4.4).

B.10 Seeds, warm-up, and repeat counts (as implemented)

- **Determinism seeds.** All timing scripts fix `torch.manual_seed(0)`. Generation/sampling uses **seed 42** (`random.Random(42).shuffle` for the GSM8K pool ordering, `e1_cot_arm_V1_n319ext.json` records `seed_sample: 42`). *Source: paper Sec. 4.1, `e1_cot_arm_V1_n319ext.json`.*
- **Warm-up / repeat.** Per-experiment warm-up and repeat counts are recorded in each timing script (not centrally). Concrete recorded values: **NREP = 5** for the official-kernel end-to-end (`official_kernel_e2e_NEW.json`) and the hardware fingerprint (`fp_stability.json` reports 5 sampling points). The hardware fingerprint uses a 3-repetition median. Waterfall and reuse figures are replay means. Cross-session external check: the graph-caliber ladder re-run on a second host in a fresh session reproduces all three levels within 1.5% (8.125 / 6.967 / 6.200 ms vs. 8.042 / 7.019 / 6.291 ms, 3-run spread $\leq$1%).
- **Discipline.** Every ratio is measured within a single session on one host. Eager-class absolutes are labeled "same-generation measurement" and are not carried across machines. *Source: paper Sec. 4.1 and App. G.*

B.11 INT8 scale / zero-point computation

Per `quant/rtn.py` (release): weights are packed as W8A16 with **group size 128**, **integer asymmetric zero-point**, **no rotation**. Scales and zero-points are computed in the quantization pass over the weight tensor within each group (full range preserved, no outlier clipping). Activations stay bf16. Only weights are dequantized at GEMV time. *Source: paper Sec. 3.4 and release `quant/rtn.py`.*

B.12 Wikitext split

The wikitext corpus is used both for the stability sweep (200 segments, teacher-forced) and as the AWQ calibration source (128 samples $\times$ 512 tokens). No held-out/test partition of wikitext is separately defined for fidelity. The 133k-token fidelity set is a distinct larger corpus used for the ΔNLL/ΔPPL caliber matrix (B.4). *Source: paper Sec. 4.2 and Sec. 4.4.*

B.13 Cross-machine reproduction protocol

Recorded in Sec. 4.1 and App. G: reproduce the **ratios** (graph-caliber ladder 8.042 / 7.019 / 6.291 ms with same-caliber denominators 8.707 / 9.765 / 12.186 ms and the 1,689.8 GB/s anchor), not the eager-class absolutes (16.752 / 12.657 ms). The second-host retest reproduced all graph-caliber levels within 1.5% and the bandwidth anchor within 0.33%. The eager-class path moved 17.7%. Before any re-measurement on new hardware: warm up the device and record the clock state (microsecond-class latency is clock-state sensitive, App. G.4). *Source: paper Sec. 4.1, `audit_results/hw_fingerprint_full_20260920.json`, `fp_stability.json`, App. G.*

Model	act recipe	deep-band r999	live-row rate	Verdict
ProSparse-LLaMA-2-7B	FATReLU	**0.390**	2.6%	✓ regime
ProSparse-LLaMA-2-13B	FATReLU	**0.647**	6.7%	✓ regime (stronger)
ReluLLaMA-7B	ReLU	0.0	51.0%	×
SparseLLM-reglu-5B	ReLU	0.0	58.3%	×
SparseLLM-swiglu-5B	SiLU	0.0	—	×
ProSparse-MiniCPM-grfull	FATReLU 0.03	0.0	15.0%	×
MiniCPM-grclust l002/l005/l01	FATReLU 0.03	0.0	14.9–15.3%	×
TurboSparse-Mistral	ReLU	0.0	—	×
Llama-2-7B / 13B dense	SiLU	0.0	—	× (control)

B.14 MoE ROUTING, A MINIMAL INSTANCE

A minimal extension of the granularity rule to MoE routing, measured on OLMoE-1B-7B-0924 (64 experts, default top-k 8, bf16, batch=1 decode, on the same RTX 5090 host as Sec. 4, wikitext prompts, `torch.manual_seed(0)`, eager timing with CUDA-event medians over 95 steps). **Latency arm (the positive answer).** Decode latency is affine in the expert count k to first order, k ∈ {1, 2, 4, 8, 16}: 87.7 / 96.7 / 98.2 / 102.5 / 109.3 ms/token (3-rep medians, per-k spreads ≤1.1% except two isolated noise bursts at k = 2 and k = 4), least-squares fit ≈91.5 ms/tok + 1.2 ms × k, R^2 = 0.84. The k-independent frame (attention, router/detection, glue, launch) dominates exactly as the fixed-cost rule expects, and the router's decision cost sits inside that frame rather than scaling with k. *Cross-run note: two earlier runs of the same arm read 84.5 / 89.2 / 91.0 / 94.0 / 100.5 and 88.6 / 98.2 / 107.4 / 105.8 / 110.9 ms/token (different clock/contention states), recorded rather than hidden. A 10-repetition re-run of the same arm (same host and setup) tightens the structure: medians 82.1 / 89.4 / 91.1 / 94.3 / 101.1 ms/token with per-k spreads of 1.2–4.8% (k = 4 and k = 8 the tightest), fit ≈85.0 + 1.06 k, R^2 = 0.865, and the same cross-entropy pattern (+152% / +58% / +7% / +1.6% against the default's 2.9033). Within any single session the k-independent frame therefore dominates the per-expert term by roughly an order of magnitude, while the absolute level moves with clock/contention state exactly as Sec. 4.1's drift note describes. A router-share profiler arm was attempted, its instrumentation did not register in this environment, and it is omitted rather than reported.* **CE gate.** The lowered-k settings degrade cross-entropy beyond the 2% gate (+152% / +58% / +7% against the default's 2.9033), while k = 16 stays within it (+1.6%). **Observation arm (pre-registered as table-only).** The temporal regime is absent at the default gate (deep-band r999 = 0.000 at k = 8 and k = 16) and appears only under the degenerate k = 1 setting (0.404 at CE 7.32, outside the operating regime), so it is tabulated and not promoted to a claim. *Source:* `iclr27_scripts/iclr27_moe_latency3.py` → `audit_results/iclr27_moe_latency3_20260923.json`, *observation arm from* `audit_results/iclr27_moe_topk_20260923.json` *(r999 values transcribed from* `iclr27_moe_topk.log` *of the main run).*

C FULL NEGATIVE-RESULTS LIST WITH VERDICT EVIDENCE

This appendix is the complete verdict ledger behind Sec. 4.5 and Sec. 5.4(i). Each excluded direction is paired with the measurement that priced it out.

C.1 CROSS-MODEL SWEEP STATISTICS (TWELVE MODELS)

Unified protocol (txt_ns, no BOS, 24 segments × 96 tokens, deep band = last 12 layers). *Source:* `appendix_G_H_tables_v213.md` *(F.3) and* `audit_results/regime_sweep1_minicpm.json`, `regime_sweep2_*.json`.

Note: the live-row rate column reports each model's first-cell (lowest-threshold) value. It is the same source as Sec. 4.4 (`regime_sweep1_minicpm.json` / `regime_sweep2_*.json`*). The MiniCPM-grclust row spans 14.9–15.3% across its l002/l005/l01 variants at the same first-cell convention as the other rows of this table.*

Every layer of every non-regime model reads r999 = 0 (no partial-depth availability). The regime appears in the ProSparse-LLaMA-2 checkpoints at live-row rates of 2.6%–6.7% under the sweep's own probe protocol, while the MiniCPM checkpoints sharing the FATReLU activation read zero.

Every zero-exceedance model sits at ≥14.9% live rows. Recipe and live-row rate co-vary across this sweep, and the threshold-forcing controls (App. C.6) isolate the separating coordinate: the ProSparse-LLaMA-2 training configuration rather than the live-row rate alone.

C.2 Prefix cross-check per-cell detail

See App. B.8 for the six-cell matrix. Summary verdict: the precise tail is construction-sensitive — it exists only under no-prefix input. Prepending one token (or BOS) collapses it at 7B without recovery over a 256-token window, while at 13B the collapse is BOS-specific and non-BOS prefixes leave the regime intact (App. F.6). Dense controls are indifferent. This bounds the mechanism's deployment relevance to generation that begins directly with content tokens (Sec. 4.4, Sec. 8(i)). *Source:* `negative_results_temporal_redundancy_20260918.md`.

C.3 13B reproduction and dose sweep

- **Reproduction.** Under the same protocol, ProSparse-LLaMA-2-13B measures a deep-band r999 of **82.1%** (mean cos 0.9991), ≈1.8× the 7B rate (46.4%, mean 0.9692). Same-size dense controls at both scales measure zero exceedances across three probe protocols. *Source: paper Sec. 4.4.*
- **Dose sweep (amplification).** Mirroring the 7B m-sweep on 13B (m = the runtime FATReLU threshold, 0.01→0.4): r999 rises **0.665** → **0.706** → **0.844** → **0.938** → **0.985** while CE stays flat (~9.6 nats under this sweep's no-BOS protocol), i.e. the response strengthens with scale and is "freer" on the larger model. A four-form fit prefers the same shape on both scales, a saturating exponential r999(m) = a − b·e^(−cm) (best AIC on 13B and 7B, R^2 = 0.972 / 0.982, plateau a ≈ 1.01 / 0.63 and rate c ≈ 7.6 / 1.9, with broadly-determined intervals at five points per scale), and the same form is preferred on the eleven-point replication set (R^2 = 0.991). *Source:* `audit_results/iclr27_induction_sparse13b_ind.json` *(13B dose) and* `audit_results/iclr27_m_sweep_20260919.json` *(7B), fitted by* `iclr27_scripts/iclr27_r999_m_fit.py` → `audit_results/r999_m_fit_20260923.json`. *The C.1 boundary matrix and these dose sweeps come from different probe families (sweep2 vs. induction/m-sweep), and values are never merged across the two.*

C.4 FFN-off artifact controls

The whole-layer FFN-skip arm (V3b) is the artifact control for "is the speed gain real or an FFN-off artifact?": under the identical fidelity protocol it gives **15.5%** paired flip and chain completeness collapse (0.51→0.19). It does not pass, so the stack's fidelity is not an FFN-off artifact. *Source: paper Sec. 4.2, Sec. 4.5.*

C.5 Reference-anchored gate negative result (B4)

An alternative gate — `cos(x_t, x_@refresh)` (anchor the decision to the last refresh rather than the lagged previous step) — was tested as a mechanism improvement. Verdict: **dominated on both axes**. It covers *less* (0.789→0.727) and the isolated-step KL is *larger* than the lagged-cosine gate. Direction excluded. *Source:* `progress_brief_20260920_reuse_fidelity_boundary.md` *(B4) and* `audit_results/reuse_boundary_b4.json`.

C.6 Threshold-forcing failures on non-regime models

Forcing the FATReLU threshold on models outside the regime does not induce it. It induces a collapse artifact instead:

- **ReluLLaMA-7B** (ReLU recipe): r999 stays **0.0** for thresholds 0.0→0.5, but cross-entropy climbs 3.10 → 8.64 at m = 0.5 — a "FFN-off" collapse (≈all dead), not a genuine stability regime. The precise tail is a LLaMA-family *collapse artifact* at extreme thresholds, cleanly excluded by the (CE-collapse + near-total-death) criterion. *Source:* `audit_results/regime_sweep2_ReluLLaMA-7B.json`.

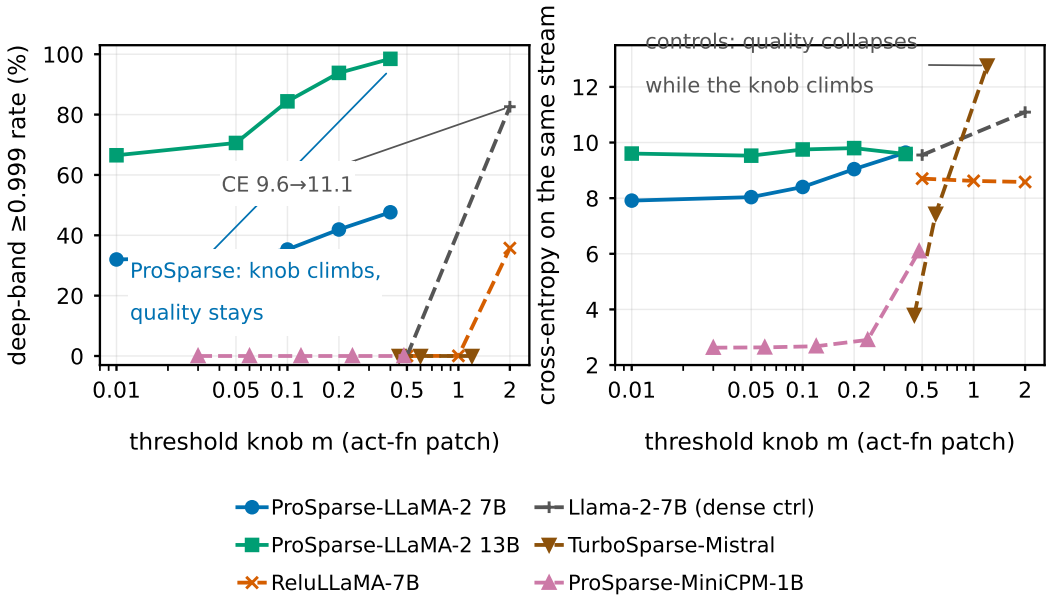

Figure 4: The threshold knob and its exclusivity, six models under the induction/m-sweep protocol. (a) Deep-band ≥0.999 rate against the knob m: the ProSparse family responds monotonically (7B 32.0% → 47.6%, 13B 66.5% → 98.5% across m = 0.01–0.4), while TurboSparse-Mistral and ProSparse-MiniCPM-1B stay at zero. (b) Cross-entropy on the same stream: the ProSparse response stays within 7.9–9.8, while exceedances elsewhere are bought at cost (Llama-2-7B 82.6% at CE 11.09, ReluLLaMA-7B 35.7% at CE 8.58 against 2.23 unpatched). All points are runtime threshold-patched settings from the same sweep.

- **SparseLLM-reglu-5B** (ReLU): r999 **0.0** across thresholds 0.0→0.5 with flat CE (~5.3) — no response at all. *Source:* `audit_results/regime_sweep2_SparseLLM-reglu-5B.json`.

The mechanism is therefore specific to the ProSparse (FATReLU) recipe. Induction attempts on other recipes either fail or collapse.

The six-model view (induction/m-sweep protocol, Fig. 4). The runtime threshold sweep across the wider family shows both halves at once. The ProSparse response is monotone in the knob, with cross-entropy within 7.9–9.8 nats at both scales (this protocol carries no BOS, while the BOS-amplifier family reads 3.34–3.55, App. F.4) as the rate climbs (7B 32.0% → 47.6%, 13B 66.5% → 98.5% across m = 0.01–0.4). The four control recipes under the same protocol either never respond (TurboSparse-Mistral 0.0% at m = 0.45–1.2, ProSparse-MiniCPM-1B 0.0% at m = 0.03–0.48) or reach exceedances only at the extreme threshold at a cross-entropy cost: Llama-2-7B measures 82.6% at m = 2.0 with cross-entropy rising to 11.09, and ReluLLaMA-7B measures 35.7% at m = 2.0 with cross-entropy at 8.58 against its 2.23 unpatched. The knob exists, and it is recipe-specific. This family (runtime threshold sweep, no retraining) is the dose-sweep family of C.3 and is never merged with the sweep2 controls above. *Source:* `audit_results/iclr27_induction_relullama.json`, `audit_results/iclr27_induction_llama2_ctrl.json`, `audit_results/iclr27_induction_ts_push.json`, *and* `audit_results/iclr27_induction_mcpm.json` *plus the C.3 dose sources, rendered by* `figures/figs_v4.py` → `fig7_knob_v3.pdf`.

C.7 False-negative injection dose-response

To price the predictor-FN failure mode directly, reuse is forced at synthetic false-negative rates r ∈ {0, 0.01, 0.05, 0.10} on GSM8K 4-shot, n=100, max_new 512. *Source:* `audit_results/iclr27_e6_fn_inject_20260919.json` *and paper Sec. 5.4(i).*

Forced FN rate r	acc	chain compl.	paired flip	n_forced (total)
0% (gate only)	0.170	0.510	0.00	0
1%	0.150	0.490	0.02	474
5%	0.140	0.390	0.11	2,485
10%	0.100	0.340	0.15	4,802

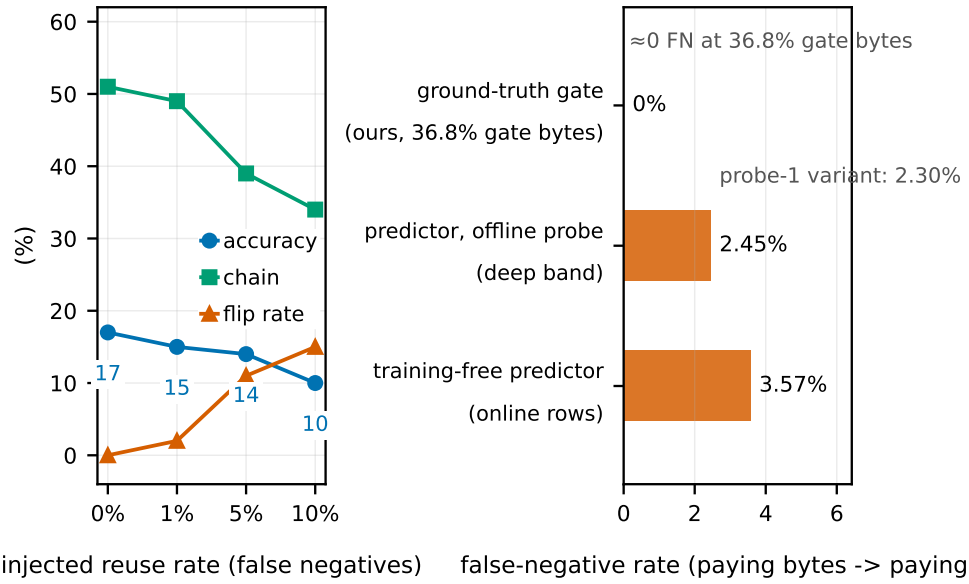

Figure 5: The information price of the reuse decision. (a) Forced false-negative dose-response: accuracy, chain completeness, and paired flip rate against the injected FN rate (0 / 1 / 5 / 10%) on GSM8K 4-shot, n=100. (b) The same axis with real predictors: the ground-truth gate pays 36.8% of weight bytes for ≈0 FN, while the training-free predictor's false-negative rates (deep-band probe at 2.45%, an earlier probe at 2.30%, online rows at 3.57%) land between the 1% and 5% rungs of the injection curve.

A real, training-free predictor in the loop (history-copy union-of-4, granted its most favorable signal) drops **1.18%** of deep-band live rows offline and **3.57%** online, and the greedy chain leaves the full-compute path at token 1–3 in **44 of 48** prompts, with **71%** of 12,288 generated tokens differing over 256 steps. The 36.8% gate tax buys exactly the absence of that divergence. The measured online rate (3.57%) lands between the 1% and 5% rungs of the injection curve above, so the predictor's failure is priced on that same axis, and the same probe settles the deep band's composition: 99.8% dead at 0.2% live rows, the direct measurement behind Sec. 6.1's migration claim. Together, the injection curve and the real-predictor control bracket the predictor-FN failure mode from both ends (Fig. 5). *Source: paper Sec. 5.4(i) and* `audit_results/predictor_probe.json` *(deep_live_frac_mean = 0.00199, recomputed), and* `figures/figs_v4.py` → `fig8_pricing_v3.pdf`.

C.8 Sec. 4.5 remaining items, and the exclusion list read as a rule — verdict evidence

The rule that unifies them (Sec. 4.5): each was excluded by the same ledger entry the design rule reads — index-compressed live rows pay a density penalty. Block-level skipping misses most opportunities. L2 pinning has no working set. Int4 spends its levels on a heavy tail. A per-layer gate spends more on decisions than it saves. The design rule is an elimination procedure first, and only then a description of the stack.

The neighborhood. In autoregressive decoding, GRIFFIN selects FF experts once per sequence, LoRA-Drop (Rajabzadeh et al., 2026) reuses layer outputs on a fixed schedule with trained corrections, RecycleGPT (Jiang et al., 2023) recycles previous states through a small module, and token-gating

Direction (Sec. 4.5)	Verdict	One-line evidence
Per-layer split graph + per-layer gate	excluded	App. D (0.8045×)
Naive int4 RTN	excluded	KL 0.319 (Sec. 4.2)
L2 persistence / pinning	excluded	working set 7.84 GB ≫ 96 MB L2, streamed weights have no temporal locality
Index / compressed live rows	excluded	at 6.79% live-row rate, metadata + gather ≥ reading coalescing-friendly 8 KB rows directly, gather roofline 904.9 GB/s @1% vs. 1,659 GB/s in-place (cross-experiment, `tf_bitwise_probe_20260920.json`)
Low-rank gate (uncalibrated)	excluded	the zero-false-negative contract admits no approximation
Intra-layer qkv fusion	not new	standard in decode engines since 2022, no new bytes
S1 custom GEMV	useful, not new	1.164× measured, component only, not a contribution
up/down access tuning	excluded	marginal bandwidth already 98.2% of sustained

lines decide per step how much to execute (Laitenberger et al., 2026). Masked-diffusion decoding exploits denoising-step redundancy: dLLM-Cache (Liu et al., 2026) caches and recomputes a similarity-selected fraction of tokens, the dKV-Cache (Ma et al., 2025), Fast-dLLM (Wu et al., 2025), Elastic-Cache (Nguyen-Tri et al., 2025), DyLLM (Lee et al., 2026), d^2Cache (Jiang et al., 2026), SparseD (Wang et al., 2025) family layers refresh triggers on it, Future Lens (Pal et al., 2023) grounds the cross-token intuition, and prompt-side caching (Prompt Cache (Gim et al., 2024), CacheBlend (Yao et al., 2024), Pensieve (Yu et al., 2023)) reuses prefixes across requests.

Ceiling taxonomy (method note). Any "limit" claim must name whose value anchors it, which layer it lives on, and whether it is hard, soft, or illusory: **L0 implementation**, **L1 algorithmic**, **L2 mathematical**, **L3 architectural**, **L4 informational**, where hard = device peak, soft = a chosen formula (e.g., the trained 6.79% live-row rate), and illusory = a framework artifact that vanishes on contact. Mistaking an L0 value for an L1 limit is the most common source of phantom ceilings.

The four questions asked of any acceleration method (expanded version). (i) Which bytes does it save? (ii) By what information does it decide those bytes can be skipped? (iii) What does that information itself cost? (iv) How is fidelity proven? Existing methods separate cleanly: ProSparse pays ground-truth information (36.8% tax, zero false negatives). TEAL, DejaVu, and CATS buy cheaper information with false negatives. FFN-SkipLLM uses a spatial prior. Quantization answers a different axis (bits per byte). Our temporal reuse uses the previous step's outputs, already resident on the device, at ≈zero tax.

Calibration and negative-result notes. On per-step equation accuracy, int4 (0.818) beats dense (0.782) only because it generates 1072 equations, 59% more than dense's 675, inflating the denominator. The excluded directions each trace to the same ledger entry the rule reads: a density penalty, missed block opportunities, an unpinnable working set, int4 levels on a heavy tail, and per-layer costs exceeding savings. By injection, forcing reuse at false-negative rates of 0/1/5/10% also moves chain completeness 0.510 → 0.490 → 0.390 → 0.340.

D SPLIT-GRAPH ENGINEERING DOSSIER

This appendix documents the per-layer-gate ablation (Sec. 4.5) at engineering level. The full per-layer refinement of the dual-graph design splits the step into a shallow-20-layer graph and a deep-12-layer per-layer eager gate. *Source for all of D: paper Sec. 4.5. Released split-graph artifact (`e3_reuse_graph_split.py`, `deep_step.py`). `E3_split_20260916.json`/ `e3_split_20260916_final.json`/ `e3_split_selfcheck.json` (server).*

Configuration	ms/token	vs. single-graph A
Single graph A (full compute)	11.166	1.00×
Split base (shallow graph + deep eager)	12.045	0.93×
Split + per-layer gate reuse	**13.878**	**0.8045× (slower)**

D.1 T1 — SHALLOW-GRAPH BITWISE ACCEPTANCE

The shallow graph (layers 0–19, full compute) is captured and replayed bitwise-identical to the engine's eager forward: **gap = 0 over 72 positions**. The KV continuity across the split boundary is likewise bitwise (KV cached continuously for 16 consecutive steps × 12 layers, zero difference). These two bitwise checks certify that splitting the graph does not perturb the shallow path.

D.2 T2 — PER-LAYER GATE

The deep 12 layers are removed from the graph and instead executed eagerly, one gate decision per layer (cosine of the lagged deep hidden state ≥ 0.9995 per layer). This is the refinement that adds a fixed decision cost at every layer.

D.3 T3 — ASSEMBLY

Three configurations are timed on the same host, same inputs:

D.4 T4 — THE 0.8045× RESULT AND ITS DIAGNOSTICS (RESIDUAL CLOSED BY REPLAY)

The per-layer-gate configuration is **slower** than full compute, a net penalty of ≈2.7 ms (13.878 − 11.166). The component-level account:

- Deep per-layer eager costs **533 μs/layer** vs. **333 μs/layer** inside a captured graph (+2.4 ms/step of launch tax).
- **12 gate-bit readbacks** cost **+1.8 ms/step** of synchronization.
- The **24.5%** layer skip this unlocks saves only **0.3–0.5 ms**. The readbacks and decisions sit outside the captured graph by construction: an in-graph decision variant was not built or measured, so this ablation prices the host-mediated design rather than the granularity idea itself, which remains untested in-graph.

Summing the component account (deep eager +2.4 ms, 12 readbacks +1.8 ms, skip savings −0.3 to −0.5 ms) gives ≈**3.8 ms**, which **does not close** against the measured incremental penalty of 1.83 ms for the per-layer-gate step (13.878 − 12.045) and the 2.7 ms total penalty. **This residual is reported as measured:** the component sum and the measured delta disagree by roughly 1–2 ms. The most likely unreconciled term is the interaction between eager launch tax and readback serialization at the 12-layer granularity (the component sum treats them as independent additions, but the measured step carries them on a shared critical path with partial overlap). The conclusion the section draws — that refining reuse granularity one notch below the 12-layer block adds a fixed decision cost exceeding the proportional savings — stands on the measured 0.8045× and the signed diagnosis above.

Closure by controlled replay (2026-09-23). A same-session three-arm experiment first reproduces the frozen arms (11.123 / 11.996 / 13.901 ms/token against 11.166 / 12.045 / 13.878, drift ≤0.5%, with skip rate and KL statistics reproduced to the digit), then isolates the decision path: replaying the live gate's own decisions with zero synchronization and no cosine recomputation leaves the trajectory bitwise identical (argmax mismatch 0, sampled max $|\Delta\text{logit}| = 0.0$ over 1,536 tokens) and measures **11.992 ms/token** against the live arm's **14.163**, Δ = **2.171 ms/token**. The replay-session live arm reads 14.163 at the upper median against the frozen arm's 13.901, a 1.9% offset that sits inside the per-segment spread of the same runs (13.5–14.4 ms), and the Δ below is computed within the replay session, where live and replay share conditions and consume identical decisions. The replay lands on the no-gate baseline (11.996), so the per-layer gate's entire increment is synchronous decision cost on the critical path, and the two component terms above are not independent addends: they share that one stalled path, which is why the 3.8 ms sum

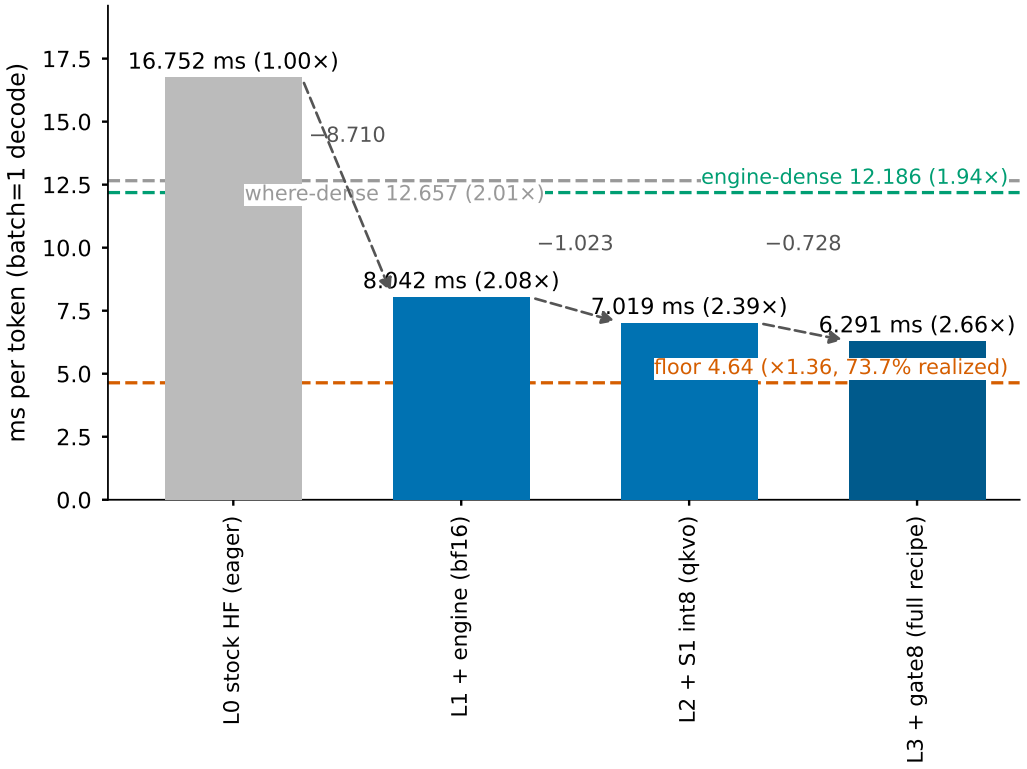

Figure 6: Waterfall (y = ms per token) of the four same-stack measured levels of Sec. 4.3, plus three dashed reference lines: the engine-dense graph stack (12.186 ms, "1.94×", matching caliber), the where-fixed dense eager stack (12.657 ms, "2.01×", eager caliber), and the bf16-dense ledger floor (4.64 ms, "gap ×1.36").

over-counts against the measured ~2 ms. An independent profiler run (ratios only, under its own overhead) matches the shape at ≈14 host sync points per token with a single graph launch per step. The residual therefore closes as an attribution rather than a to-the-decimal account: the gate's measurable cost is the synchronous decision path, and the decision wait is the measured driving term. *Source:* `iclr27_scripts/iclr27_splitresidual_20260923.py` → `audit_results/iclr27_splitresidual_20260923_{a0,a1,a2,merged}.json`. *Baseline consistency:* full-compute single-graph measurements across same-session contexts are 11.142–11.166 ms (0.2% apart). The same engine's eager path measured 14.229 ms in an earlier session, versus 11.728 ms under a whole-step graph (s7c record).

Caliber accounting for L0→L1. Against the official graph path (8.707 ms), 16.752/8.707 = 1.92× is the pure caliber jump, 8.707/8.042 = 1.08× our engine's same-caliber gain. Three labeled denominators: (a) matching graph caliber, primary: the dense arm (all rows, whole-step graph) is 12.186 ms, so 1.94× (row-skip 1.52× × int8+gate 1.28×), sparse arm re-measured at 8.150 ms in the same session. (b) eager caliber: the `torch.where`-fixed dense is 12.657 ms, so 2.01× (the graph removes launch tax, not math). (c) download denominator: 2.66× against the L0 form users download today.

Floor decomposition. The distance to the 4.64 ms floor is dominated by S12 non-weight overhead (1.382 ms, ≈0.431 ms avoidable) and S3 attention.

System / arm	Caliber	ms/token	Same-caliber relation	Fidelity	Source
Official release form (download-and-run, eager)	eager	**16.752**	—	—	s7c_clean_-e2e
Official kernel path (eager)	eager	13.45	≈1.02× over hand-written dense (13.695)	gate-exact	official_kernel_e2e
Official kernel graph-capture path	graph	**8.707**	—	gate-exact	official_kernel_e2e
TEAL (same-model port, graph path)	graph	**9.765**	—	tolerance-type (magnitude)	teal_e2e
Our L1 = engine bf16	graph	**8.042**	1.08× official graph, 1.21× TEAL graph	exact (bitwise replay)	engine_-dense_-graph / s7c
Our L2 = +int8 recipe	graph	**7.019**	—	KL 0.00863, ΔNLL CI crosses 0	e2e_int8_v4
Our L3 = gate8 (+row-skip + dual-graph)	graph	**6.291**	**1.38× official graph, 1.55× TEAL graph**	flip 2.0% (n=200), self-consistent within 0.2%	engine_-dense_-graph
Our e5g8 (control arm)	graph	6.618	—	—	e2e_int8_v4
Strongest dense stack (ours, same kernel structure, where-dense)	graph	**12.186**	**1.94**× (gate8 ÷ dense stack)	—	engine_-dense_-graph
where-fixed dense (eager)	eager	12.657	2.01× (cross-caliber, disclosed)	—	s7c_clean_-e2e
FFN whole-layer skip 12 layers (V3b speed arm)	graph	8.777 (upper bound 1.266×)	—	**COLLAPSE**: flip 15.5%, chain 0.51→0.19, greedy agreement 0.02	ffn_skip12_-e2e + cot_arm_-V3b
(Reference) official release retest on second host (10771)	eager	20.433	1.52× (vs its kernel eager)	—	official_kernel_e2e

E COMPETITOR END-TO-END TABLE

We report the full end-to-end ladder and the head-to-head competitor numbers on one hardware/protocol, each annotated with its execution caliber, so that no reader has to reconstruct the caliber bookkeeping from Sec. 4.3.

Caliber discipline. Every cell shares the model (ProSparse-LLaMA-2-7B), the card (RTX 5090), batch = 1, and the protocol (P = 64 prefill + generation-segment wall-clock timing). The three execution calibers in play are: *eager* (stock PyTorch execution, per-layer boolean-mask synchronization, no CUDA graph), *graph* (whole-step CUDA graph capture), and *release form* (the official repository's download-and-run artifact, which ships eager-only). The headline claims are stated at matching calibers. Cross-caliber ratios are disclosed separately and never combined with same-caliber ratios.

E.1 TABLE E.1. END-TO-END LADDER AND COMPETITORS (GRAPH CALIBER PRIMARY)

Caliber footnote (carried from Sec. 4.3). The four headline figures used in Sec. 4.3 / Abstract are: **1.38**× (gate8 vs official *graph* path), **1.55**× (gate8 vs TEAL *graph* path), **1.94**× (gate8 vs our strongest measured dense stack, same *graph* caliber), and **2.66**× (gate8 vs the official *eager* release form, cross-caliber). The L0→L1 step crosses calibers (eager→graph): against the official graph path (8.707 ms), 16.752 ÷ 8.707 = **1.92**× is the pure caliber jump and 8.707 ÷ 8.042 = **1.08**× is our engine's gain at the same caliber. The 1.94× primary dense-baseline figure factors as row-skip (12.186 ÷ 8.042 = 1.52×) × int8+gate (8.042 ÷ 6.291 = 1.28×). The 2.01× (12.657 ÷ 6.291, eager caliber) and 2.66× are peers of each other and are both labeled as eager-caliber.

Reproducibility footnote (carried from Sec. 4.1). Cross-session retest on a second RTX 5090 (host 10771): the graph-caliber L1/L2/L3 reproduce at **8.125 / 6.967 / 6.200** ms (within ±1.5%, 3-run spread ≤ 1%) against the original-host **8.042 / 7.019 / 6.291** ms. The stock/eager path drifts more (16.752 ms on the original host → 19.712 / 20.433 ms across sessions), so the release-form timing spans 16.8–20.4 ms and is always substantially slower than the stack. The original-host numbers

#	Model · band	τ	Cap	Cov.	agree	Fired-step KL [p50/p90/max]	Speedup
1	7B · deep ring (default gate)	0.9995	—	0.984	1.0	[1.12, 1.65, 1.70]	**1.230**×
2	7B · deep ring	0.9999	8	0.789	1.0	[0.144, 0.191, 0.517]	1.180×
3	7B · deep ring	0.9999	4	0.711	1.0	bimodal (≈half ≤1e-4 / half 0.1–0.5)	1.158×
4	13B · shallow ring (default gate)	0.9995	—	0.984	1.0	[0.011, 0.036, 0.066]	~1.19×
5	13B · shallow ring	0.9999	8	0.805	1.0	[0.0003, 0.025, 0.027]	**1.154**×
6	13B · shallow ring	0.9999	3	0.680	1.0	[0.00026, 0.024, 0.027]	—
7	13B · deep ring	0.9999	8	0.477	0.758	[0.0012, 0.086, 0.175]	—
8	13B · deep ring, m = 0.08	0.9999	8	0.680	0.945	[0.0002, 0.091, 0.278]	—
9	**13B · deep ring, m = 0.15**	0.9999	8	0.672	**1.0**	**[2.5e-4, 0.0995, 0.203]**	—
10	13B · deep ring, m = 0.15	0.9999	4	0.609	1.0	[0.0002, 0.0977, 0.198]	1.116×

(16.752 / 12.657 / 8.042 / 7.019 / 6.291 / 11.142) are retained as the primary caliber. The 2.66× ratio is computed from original-host numbers on the same host.

F REUSE-FIDELITY DOSSIER

This appendix decomposes and bounds the temporal reuse mechanism: the coverage–fidelity–speed frontier (F.1), the reuse-block-size axis (F.2), the twelve-model boundary map (F.3), the amplifier safe zone (F.4), the long-range / idle / negative results (F.5), the prefix recovery dynamics (F.6), and the flip-margin audit (F.7). All "speedup" figures are within-graph-caliber divisions. Every number traces to a `audit_results/` *JSON recomputed under the zero-trust ledger.*

F.1 FRONTIER TABLE (COVERAGE ⇄ FIDELITY ⇄ SPEED, ALL SAME-RUN MEASURED)

The table below is the per-model, per-setting coverage–fidelity–speed menu referenced in Sec. 4.4. "Coverage" (B) is the fraction of steps where reuse fired. "Agree" is greedy argmax agreement against the full-compute reference. "Fired-step KL" is the divergence at the positions where the gate opened.

Note. Here *deep ring* and *shallow ring* name the sweep's two repetition-profile inputs (the EOS and punctuation rings), not layer bands. The full 22-configuration sweep (including τ = 0.99999 with zero global triggering, and cap 1/2/3 fine scans) is in `reuse_boundary_b2/b3*`. The deep-band 7B default setting (row 1) reaches 1.230× because the mechanism's fixed cost sits below timing resolution once the gate is computed inside the captured graph (a 12-float asynchronous readback and nothing else on the host). The realized reduction equals the deep-block FFN share it removes (≈2.1 ms of the 11.2 ms step). On natural text, where the gate fires rarely, the same implementation measures 1.000× (0.9996× / 1.0001× across two runs) — at zero measured cost. Row 1's fired-step KL distribution belongs to this reuse-heavy regime, where fired steps sit deep in reuse chains, while the natural-text probe at the same gate measures 0.19/0.39 (Sec. 4.4, B.9). The scatter view of the same sweep, with paired timing, is Fig. 7: 160 runs in total, 57 in the zero-coverage band (among them all 18 ReluLLaMA control runs) and 103 at nonzero coverage, with measured same-session ratios of 1.076× / 1.085× / 1.135× / 1.154× for the four capped 13B settings. *Source:* `figures/figs_v4.py` → `fig9_frontier_v3.pdf`*, rendering the b1–b4 reuse-boundary families from the same sweep.*

F.2 SECOND AXIS: REUSE-BLOCK SIZE

The default reuse block is the 12-layer deep band. This axis asks how coverage and fidelity move as the block is shrunk to 8 / 6 / 4 layers. The primary 7B scan (τ = 0.9995, shallow ring) carries the full per-block fidelity:

Table F.2a. 7B shallow ring, τ = 0.9995 — coverage and fidelity by reuse-block size.

The same divergence order holds across block sizes (Sec. 4.4). The complementary 13B shallow-ring scan at m = 0.08 (full per-block fidelity, `extra_skip.json`) shows the saturation pattern — coverage saturates at 0.984 with agree 1.0 and bounded KL that *decreases* as the block shrinks:

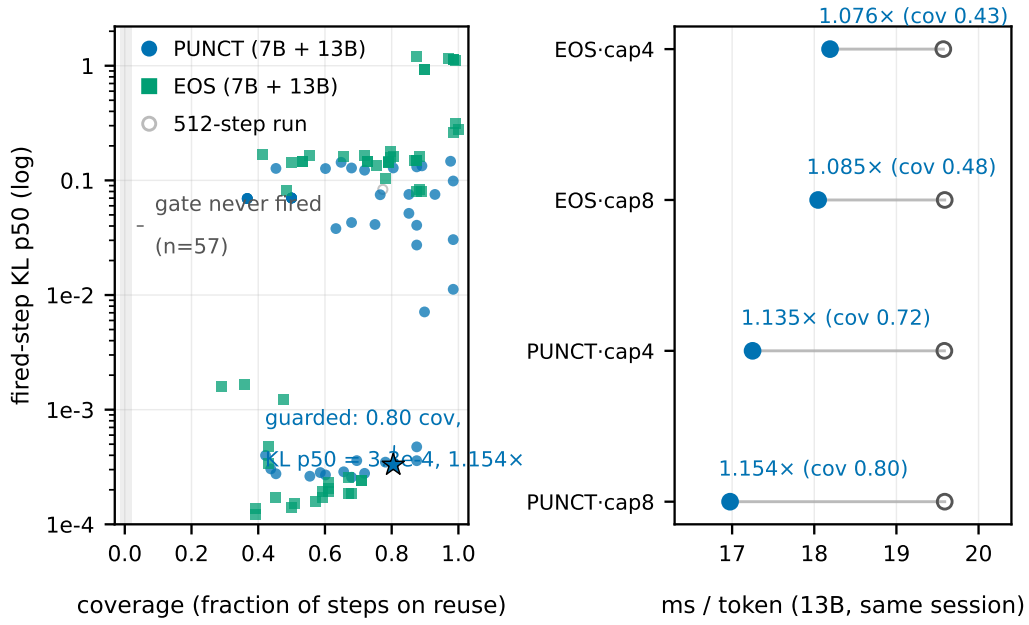

Figure 7: The operating frontier and its price. (a) Coverage against fired-step KL p50 (log scale) for every measured configuration of the τ/cap sweep. The sweep pools the b1–b4 families into 160 runs, a superset of the 136 points plotted in Fig. 8, adding the 7B default-gate sweep and the 512-step and cap-4 anchors, with 103 runs at nonzero coverage and 57 in the zero-coverage band, among them all 18 ReluLLaMA control runs (colour separates the two input regimes, pooling 7B and 13B runs). The guarded setting is starred. (b) Paired base / reuse milliseconds per token for the four capped 13B settings, same session, with measured ratios 1.076× / 1.085× / 1.135× / 1.154×.

Reuse-block size (deep layers)	12	8	6	4
Coverage B	0.367	0.938	0.969	0.984
agree	0.79	0.74	0.63	0.80
Fired-step KL_max	0.221	0.297	0.290	0.241

Table F.2b. 13B shallow ring, act-thr 0.08, repetition-loop (PUNCT) regime, default gate ($\tau = 0.9995$, cap 0).

Note. The 7B $\tau = 0.9995$ scan now carries the full per-block fidelity (agree 0.79 / 0.74 / 0.63 / 0.80 and fired-step KL_max 0.22 / 0.30 / 0.29 / 0.24 for blocks 12 / 8 / 6 / 4): coverage saturates with finer blocks, while the agreement stays below the capped 13B arm's 1.0, consistent with the punctuation ring carrying the higher divergence of App. F.1. Speedup per block size is not reported for either scan. *Source (7B arm):* `audit_results/extra_skip_7B_20260924.json` *(re-run 2026-09-24 under the same protocol, with coverage reproducing the earlier scan to within 0.001).*

F.3 Coverage matrix (12 models, uniform protocol)

Uniform protocol: `txt_ns` without BOS, 24 segments × 96 tokens, deep band = last 12 layers. `act` is the activation function. `Deep-band r999` is the fraction of adjacent-step pairs with cosine ≥ 0.999 in the deep band. `Live rate` is the measured activation liveness.

Layerwise check. Every non-regime model reads r999 = 0 at *every* layer → no "partial-depth" usability at all.

Live-rate note. The live-row rates in this table are taken from each model's first (lowest-threshold) cell in `regime_sweep1_minicpm.json` / `regime_sweep2_*.json`, the same source as Sec. 4.4 — hence the consistent ≥14.9% band (14.9–15.3%) quoted in Sec. 4.4.

Reuse-block size (deep layers)	12	8	6	4
Coverage B	0.984	0.984	0.984	0.984
agree	1.0	1.0	1.0	1.0
Fired-step KL_max	0.105	0.102	0.065	0.062

Model	act	Deep-band r999	Live rate	Verdict
ProSparse-7B	fatrelu	**0.390**	2.6%	✓ regime
ProSparse-13B	fatrelu	**0.647**	6.7%	✓ regime (stronger)
ReluLLaMA-7B	relu	0.0	51.0%	×
SparseLLM-reglu-5B	relu	0.0	58.3%	×
SparseLLM-swiglu-5B	silu	0.0	—	×
ProSparse-MiniCPM-grfull	fatrelu 0.03	0.0	15.0%	×
MiniCPM-grclust-l002 / l005 / l01	fatrelu 0.03	0.0	15.3 / 15.2 / 14.9%	×
TurboSparse-Mistral	relu	0.0	—	×
Llama-2-7B / 13B dense	silu	0.0	—	× (control)

Protocol note (alignment with Sec. 4.4). The 82.1% 13B deep-band rate and the 46.4% 7B deep-band rate quoted in Sec. 4.4 use its deployment probe protocol (cosine $\geq$ 0.999, L20–31 average), which differs from the uniform F.3 protocol above. Under the uniform F.3 protocol the 13B deep-band r999 is 0.647 and the 7B is 0.390. These are reported on a different (more conservative) sample pool and are not to be arithmetically compared with the Sec. 4.3 headline rates. The quantity that separates the two outcomes is the recipe, not the live-row rate (App. C).

F.4 Amplifier safe zone (13B, BOS protocol)

The amplifier raises the live-row rate (m = 0.08 / 0.15 / 0.25) to push coverage into the deep loop. `R999 (no BOS)` is the deep-band exceedance rate without a leading BOS token, and `CE (BOS)` is the cross-entropy under the BOS protocol.

Note. Under the BOS protocol, r999 is always 0 (consistent with the established "prefix collapses the stable tail" boundary). The 0.15 setting is the operating point that holds CE flat (3.3401) while lifting coverage to 0.930. 0.25 Collapses (CE 5.4143).

F.5 Long-range / idle / negative results

Note. The 512-step result bounds the temporal drift: divergence grows with chain length but stays bounded across quarters, and agreement only dips in the final quarter (0.844) where the repetition loop begins to break down. The idle-cost result shows the mechanism imposes no measurable tax when the gate never opens (dense controls, and ReluLLaMA where the recipe yields zero exceedances).

F.6 Prefix recovery dynamics (256-token continuation windows)

The static boundary of Sec. 4.4(i) is resolved here by position, on both scales. Protocol: a 64-token natural-text base (the same pool for every construction) is prefilled, then 256 greedy continuation tokens are decoded, and the deep-band rate (cosine $\geq$ 0.999 over the last 12 layers) is reported over four position windows. Prefix heads are prepended to the base (chat = 22-token `[INST]` system prompt, instruction = 8 tokens, chat + BOS = 23 tokens), and no BOS is used except in the labeled chatbos construction. Model: ProSparse-LLaMA-2-7B / 13B (bf16, FATReLU 0.01, official SparseLLM weights), segments = the first 24 (7B) / 12 (13B) qualifying lines of the wikitext parquet pool, deterministic under `torch.manual_seed(0)`.

Note. At 7B every tested construction collapses the tail and it does not recover (zero in all four windows and all twelve layers, 73,440 pairs per construction). At 13B the boundary is construction-specific rather than general: BOS collapses it (the corpus-probe cross-check reads r999 = **0.000** for a BOS-prefixed build against **0.672** for the matched no-prefix probe of the same 24-segment teacher-forced protocol, and the single-token-prefixed build of the same family reads 0.919 under its own 96-token window), while the chat-templated prompt does not suppress the regime at all: the rate rises from 0.800 in the first window to 0.966 in the deepest. Without a prefix, the 7B regime decays with

thr	r999 (no BOS)	CE (BOS)	Functional-equivalence check
default (0.9995)	0.681	3.3735	—
0.08	0.834	3.5411	7 prompts × 128-step greedy continuation bit-identical
0.15	**0.930**	**3.3401 (flat, BOS protocol)**	not separately tested (the 0.08 check is shown above)
0.25	0.945	**5.4143 (collapse)**	—

Item	Result
Long-range 512 steps (13B@0.08 guarded)	Coverage 0.773, agree 0.951, KL [0.083, 0.178, 0.278], per-quarter drift [0.032, 0.061, 0.096, 0.107] (**bounded**), per-quarter agree [1.0, 1.0, 1.0, 0.844]
Idle cost (mechanism on/off)	ReluLLaMA-7B **+0.052 ms**, dense Llama-2-7B **−0.103 ms** (noise level, gate never fires)
Alternative gate (anchored cos(x_t, x_{@refresh}))	**Not dominant**: coverage 0.789→0.727, isolated-step KL larger → direction excluded
Bitwise stability	TF 768 positions/model × 12 layers (18,432 layer pairs) **zero hits**, 512-step loop whole block **zero hits**
False-negative injection (GSM8K 4-shot, n=100, max_new 512)	0/1/5/10% → acc **0.170 / 0.150 / 0.140 / 0.100**, chain **0.510 / 0.490 / 0.390 / 0.340**, flip 0.0 / 0.02 / 0.11 / 0.15

position into the window (52.6% → 31.0%) while the 13B reference rises (66.3% → 74.9%). Window definition: pair index k ∈ [0,8) / [8,32) / [32,96) / [96,255) over the continuation tokens. This probe runs its own 256-token window protocol (CW 96 → 256) and is not merged with the 24×96 numbers of C.1/C.3. *Sources:* `audit_results/iclr27_e3_20260923/probe_sparse_{plain,chat,instr,chatbos}_recover256_n24.json`, `probe_sparse13b_{plain,chat}_recover256_n12.json`, `probe_dense_chat_recover256_n12.json` *(scripts* `iclr27_scripts/iclr27_e3_prefix_recovery.py`*), plus the corpus-probe cross-check* `probe_sparse13b_txt_{fake,}_n24.json` *(distinct 96-token protocol, cited as corroboration only).*

F.7 Flip-margin audit (what the residual disagreement is made of)

Free-running greedy generation on natural text (24 segments × 95 steps, default gate $\tau = 0.9995$, no chain cap) yields pooled token-level agreement 0.924 at 13B and 0.950 at 7B between the reuse arm and full compute, and the disagreement concentrates where the model's own argmax margins are shallow: at 13B the base top1−top2 gap has median 0.078 on flip steps against 0.656 elsewhere (36.4% of flips below 0.05, against 3.6% of non-flip steps), while 7B's margins are wider overall (median 1.0). Cache aging is not the driver: the flip rate inside long gate-open streaks ($\geq$32 consecutive reuse steps) is 3.1% against 7.6% overall. The residual disagreement is therefore located in the logit tie zone rather than in long-cache aging. This probe is free-running, a stricter setting than the teacher-forced protocol of Sec. 4.4, and it is reported as a supplementary measurement: the pre-registered <3% criterion (Sec. 4.2, App. B.3) applies to the GSM8K long-chain paired-flip protocol, not to this one. For orientation, App. F.1's 13B deep-ring row (0.758, a stricter gate with an 8-step cap) is a different measurement rather than a contradiction. *Source:* `iclr27_scripts/iclr27_13b_logitgap_20260923.py` → `audit_results/iclr27_e5_logitgap_20260923.json` *(13B weight manifest hash-verified before the run).*

Fidelity tail. Under natural-text teacher forcing the mechanism posts argmax agreement 1.0 with KL median 0 and p90 0, while a matched probe shows the gate opens on 2.34% of positions with divergence 0.19 nats at the median and 0.39 at the maximum. Divergence grows with chain length (≈0.20 → 0.39 → 1.18 nats across the first/sixth/twelfth reuse steps), an isolated reuse step carries 0.07–0.19 nats, and no deep layer is bitwise identical in any of the 18,432 layer pairs: the regime is stability at the 0.999 bar.

The chain cap (detail: App. F). The guarded setting ($\tau = 0.9999$, cap 8) on 13B holds coverage 0.805 at agreement 1.0 and 1.154× end-to-end. On the 512-step horizon the guarded arm (13B at m = 0.08) holds coverage 0.773 at agreement 0.951 (per-quarter drift [0.032, 0.061, 0.096, 0.107],

Model · prefix construction	s0_8	s8_32	s32_96	s96_-256	Full r999	Full mean cos
7B · chat prompt, 22 tokens	0.000	0.000	0.000	0.000	0.000	0.828
7B · instruction head, 8 tokens	0.000	0.000	0.000	0.000	0.000	0.875
7B · chat prompt + BOS, 23 tokens	0.000	0.000	0.000	0.000	0.000	0.769
7B · no prefix (reference)	0.526	0.481	0.464	0.310	0.371	0.948
7B · no prefix, dense control (Mistral-7B)	0.000	0.000	0.000	0.000	0.000	0.495
13B · chat prompt, 22 tokens	0.800	0.906	0.959	0.966	0.953	0.9994
13B · no prefix (reference)	0.663	0.650	0.745	0.749	0.736	0.9991

Item	Value
GPU	RTX 5090 (sm_120, 96 MB L2, VBIOS 98.02.2E.00.90, driver 580.105.08)
GPU UUID	`GPU-b9e92bd3-d510-e164-2e37-6981390601e1` (re-verify on migration)
CPU / memory / OS	Xeon Platinum 8470Q ×2 sockets (208 threads) / 754 GB / Ubuntu 22.04.5 LTS
Software	torch 2.8.0+cu128 · triton 3.4.0 · CUDA runtime 12.8 · Transformers 4.41.2
Sustained read bandwidth	**1,684.3 GB/s** (paper anchor 1,689.8 GB/s, −0.33%)
L2 residency read / bf16 TFLOPS	3,202 GB/s / 238.2
Kernel launch / graph replay	6.43 μs / **4.74 μs**

agreement [1.0, 1.0, 1.0, 0.844]). On 7B the distribution stays bimodal under every cap, hence a per-model menu rather than a single median. Graph replay is bitwise-identical to the engine's eager forward (gap 0 over 72 positions), and agreement stays at 1.0 on 7B's default gate, rising with amplification on 13B (0.758 $\rightarrow$ 1.0 at 0.15).

Operating the mechanism. For a deployer the mechanism is three dials: the gate, the chain cap, and the block size. Leave the gate at the measured knee (0.9995 on the deep band), add the cap where fidelity matters more than coverage (0.9999 with cap 8), and shrink the block where the deep block never triggers.

Boundary map, in full (expanded version). (i) *Sequence start:* one prepended token collapses the stable tail at 7B (42.6% / 32.3% $\rightarrow \approx$0) and it does not recover over 256 continuation tokens, while at 13B the collapse is BOS-specific and ordinary-token and chat-templated prefixes leave the regime intact (F.6). (ii) *Models:* the regime tracks the ProSparse recipe rather than the live-row rate, strengthening with scale (13B 82.1% vs. 7B 46.4%) across twelve swept models at 2.6–6.7% live rows, while every zero-exceedance model sits in a different recipe at $\geq$14.9% (protocol matters: 2.57% teacher-forced vs. 0.29% self-generated, and the shallow band moves by less than 1.4$\times$ between protocols, 17.9% against 12.8%). (iii) *Controllability:* raising the FATReLU threshold amplifies the regime monotonically (7B 32.0% $\rightarrow$ 47.6%, 13B 66.5% $\rightarrow$ 98.5%) while the sensitive metric stays flat inside the usable range (cross-entropy stays within 3.34–3.54 nats against a 3.37 baseline to 0.15, greedy continuations at 0.08 are token-identical over 7 prompts $\times$ 128 steps, and the rate climbs to 93.0% at 0.15), with quality giving way past it (0.25 leaving the range at 5.41) and the amplification holding across the tail (0.949–0.963 from 0.45 through 0.80, tabled below). Forcing the threshold on a non-regime model does not induce the regime: ReluLLaMA-7B shows a rising mean cosine (0.49 $\rightarrow$ 0.72) at zero bar-exceedance but a collapsed cross-entropy (3.10 $\rightarrow$ 8.64), a distinguishable artifact excluded by construction (App. F).

G HARDWARE FINGERPRINT & PORTABILITY DOSSIER

This appendix records the static machine identity, microbenchmarks, per-model execution fingerprints, cross-card anchor reproduction, temporal stability, and the portability classification that lets the graph-caliber conclusions travel between machines while the eager-caliber absolutes do not (Sec. 4.1, Sec. 3.1). All figures are from `hw_fingerprint_full_20260920.json` *and the server fingerprint dated 2026-09-20.*

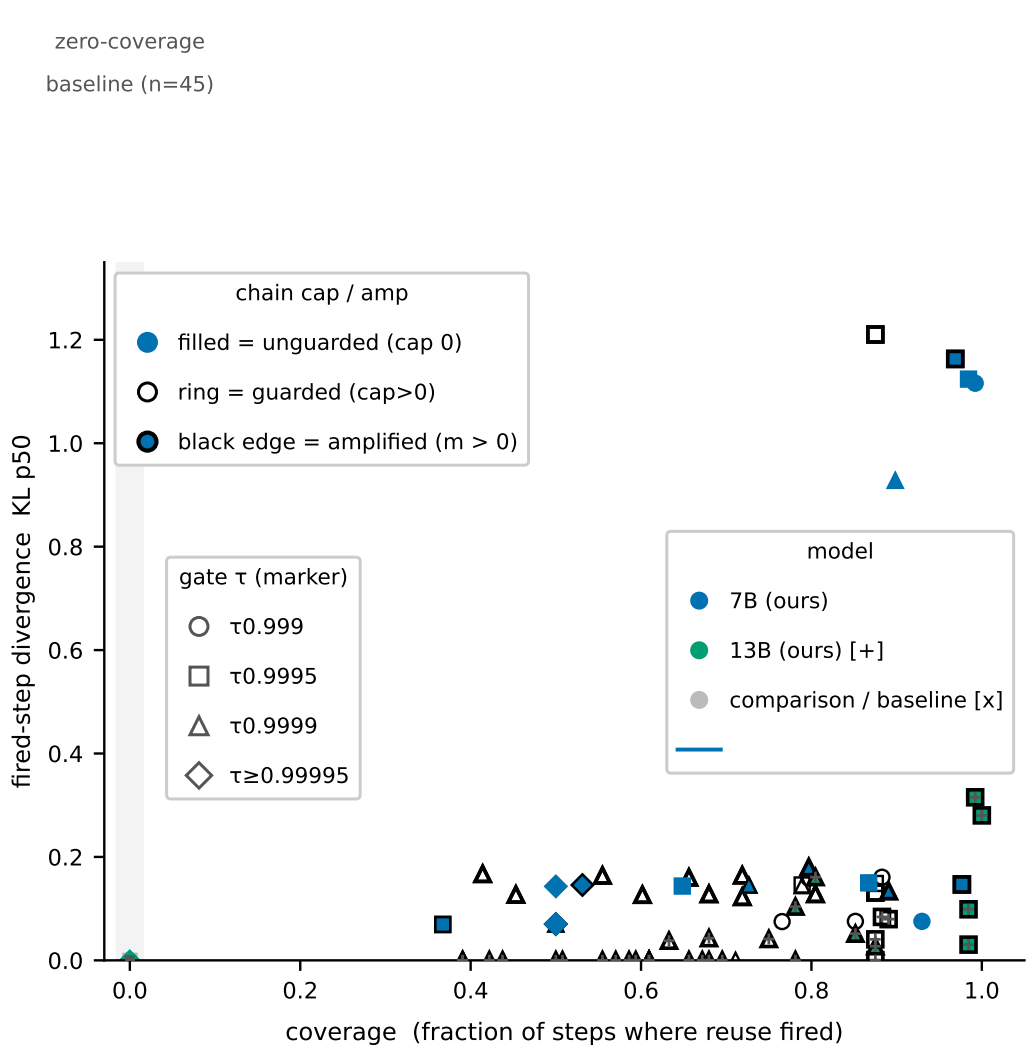

Figure 8: The measured redundancy space. Each point is one measured configuration of the reuse mechanism, placed by how far reuse is pushed (coverage, the fraction of steps on which it fired) against the divergence it incurs on those steps (fired-step KL, p50). Colour separates the 7B (blue) and 13B (green) runs from the ReluLLaMA-7B comparison (gray), marker shape encodes the gate threshold (five values, 0.999–0.99999), filled versus ringed markers separate unguarded runs from chain-capped runs (caps 1/2/3/4/8), and a black edge marks amplified-threshold runs (m = 0.06 / 0.08 / 0.15). The band at zero coverage collects the 45 runs on which the gate never fired. All 136 points are measured, from the same reuse-boundary sweep that App. F tabulates.

Model	eager	graph	reuse	reuse ÷ graph	Achieved bandwidth
7B	20.2 ms	11.29 ms	**9.14 ms**	**1.235**×	1,194 GB/s
13B	26.6 ms	19.59 ms	**16.37 ms**	1.197×	1,328 GB/s

G.1 Identity and microbenchmarks

G.2 Per-model execution fingerprint (batch = 1, 3 reps median)

Note. Reuse mechanism measured speedup: 7B 11.29→9.14 = **1.235**×. 13B 19.59→16.37 = **1.197**× (consistent with the historical record). Achieved bandwidth is measured against the sustained anchor (1,194 / 1,328 GB/s vs 1,689.8 GB/s = 70.7% / 78.6%). Graph-caliber variance is ±0.4% (7B) /

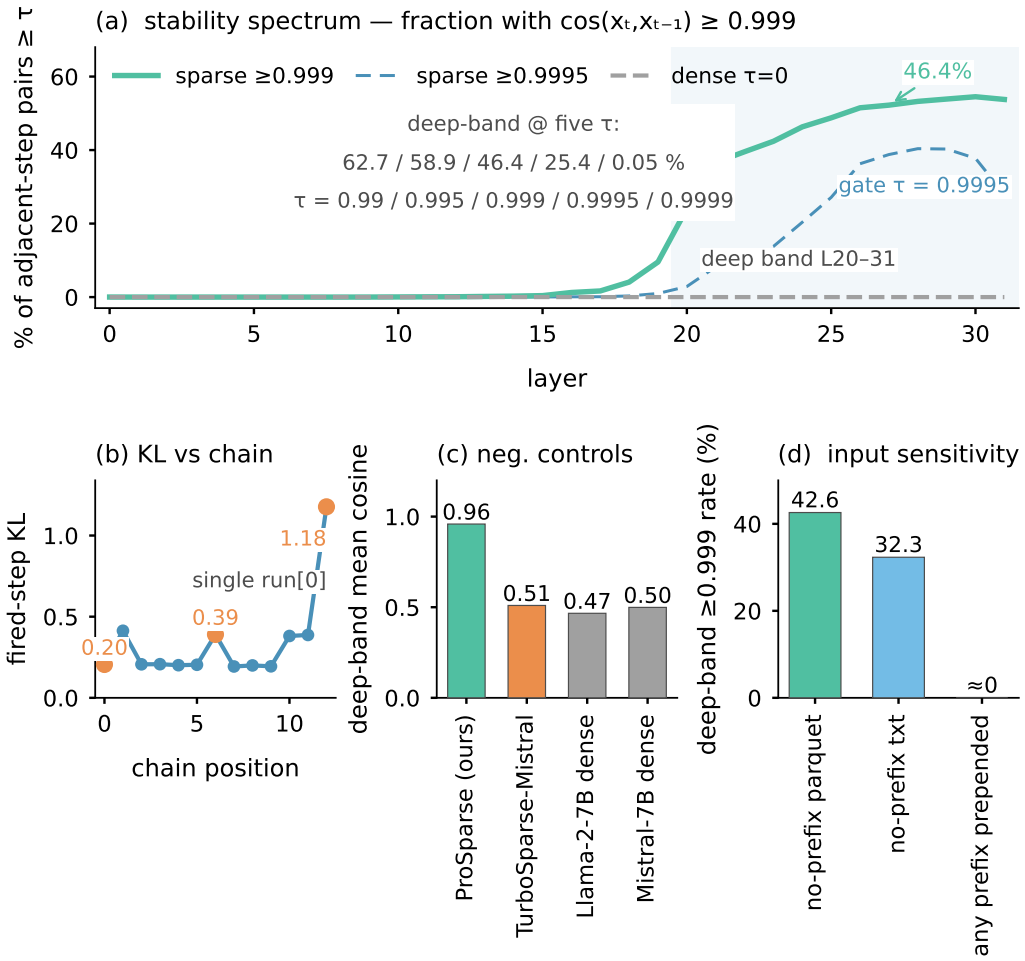

Figure 9: The stability regime and its boundaries. (a) fraction of steps with cosine(x_t, x_{t−1}) ≥ 0.999 per layer (0–31): ProSparse averages 46.4% over L20–31, above 50% in L26–31, deep-band rates at five thresholds annotated (62.7 / 58.9 / 46.4 / 25.4 / 0.05% at ≥0.99. . . ≥0.9999), a same-size dense model at zero exceedances (0/19,000 pairs per layer), gate threshold 0.9995 marked. (b) fired-step KL against chain position (7B, default gate): flat early (0.20), mid-chain bumps (0.39), sharp tail rise (1.18), motivating the chain cap. (c) Deep-band mean cosine: ProSparse 0.96 vs. TurboSparse-Mistral 0.51, dense Llama-2-7B 0.47, dense Mistral 0.50. (d) Deep-band ≥0.999 rate by input construction: no-prefix parquet 42.6%, no-prefix txt 32.3%, ≈0 once any prefix is prepended.

Anchor	Original record	This card	Deviation
Official kernel graph	8.707 ms	**8.712 ms**	**−0.06%**
TEAL graph	9.765 ms	**9.672 ms**	−0.95%
Sustained read bandwidth	1,689.8 GB/s	**1,684.3 GB/s**	−0.33%
stock/eager path	16.752 ms	**19.712 ms**	**+17.7%** (!)

±0.2% (13B). Use the median for cross-machine comparison. This table uses the reuse *test skeleton* (all-dense graphA), distinct from the released engine waterfall of Sec. 4.3 (L0–L3). The two are not to be mixed.

Sample	Sustained read GB/s	Graph replay μs	bf16 TFLOPS	Clock
k0	1,684.7	3.81	238.8	2512 MHz
k1	1,684.0	5.31	238.9	1627 MHz
k2	1,684.3	5.32	238.6	180 MHz
k3	1,683.5	5.85	238.8	487 MHz
k4	1,683.7	7.56	238.2	1117 MHz
Range	$\pm$**0.04%**	**2.0**$\times$	$\pm$0.15%	—

Class	Criterion	Re-measure after migration?
Portable (graph class, bandwidth anchors, ratios)	Cross-card / cross-session deviation < 1.5%, with composed ratios up to $\approx$2.5% (Sec. 4.1)	No (quote / convert directly)
Not portable (eager/stock caliber, absolute monotonic latency)	Card-to-card up to 17.7%, μs latency clock-dominated up to 2$\times$	**Must re-measure, do not carry over**

G.3 Anchor cross-card reproduction (migration-critical)

Note. The graph-caliber and bandwidth-class anchors reproduce within 1%. The eager/stock path varies by up to 18% *between identical-model cards*. Consequently all eager-caliber denominators in the paper (L0 = 16.752 ms, where-fixed dense = 12.657 ms) must be re-measured on a new machine and must not be carried over. The cross-session retest of the graph-caliber ladder (8.125 / 6.967 / 6.200 ms, Sec. 4.1) confirms the same travel for the L1/L2/L3 arms, within 1.5%.

G.4 Temporal stability (5 samples $\times$ 2-minute interval)

Conclusion. Large-load anchors (bandwidth / TFLOPS) are extremely stable across time ($\pm$0.04–0.15%) and are safe conversion bases. The μs-scale latency anchors are dominated by clock state (cold clock 180 MHz $\rightarrow$ 7.56 μs vs warm 2512 MHz $\rightarrow$ 3.81 μs), so the re-measurement protocol must warm up first and record clock state simultaneously. This also explains the small-metric drift between sessions.

G.5 Portability classification and usage rules

Rule. Every cross-caliber conclusion in the paper is built only on the *portable* class. Eager-class numbers are always labeled "same-generation measurement". Re-measurement protocol: warm up first and record clock state at the same time. Conversion uses the five anchors (sustained bandwidth, bf16 TFLOPS, graph replay latency, 7B graph ms/token, 13B graph ms/token). The eager/graph caliber ratio itself is machine-dependent (1.80 for 7B / 1.36 for 13B on this host) and must be re-measured. Limitation: single-card, single-host, idle-environment sampling. Multi-card / NVLink / PCIe contention not measured. Sparse kernels (official / TEAL / ours) are likely more machine-dependent than the dense anchors, so post-migration reproduction deviation may exceed the dense-anchor deviation reported here.