THE PRICE OF A MISS: SERVING-AWARE EXPERT ROUTING FOR EDGE MOE INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Expert-cache misses connect routing decisions to the cost of serving Mixture-of-Experts models under a GPU memory budget. We present TollRoute, a training-free routing rule that replaces a missing expert with a resident expert only when its router logit lies within a fixed margin of the best desired expert. The rule guarantees conditional miss non-increase and retains at least an $e^{-\gamma}$ fraction of the displaced expert's routing weight for each substitution. Its serving interface uses layer–expert cache keys, admits only experts transferred to the GPU, and calibrates the transfer/CPU split with the target platform's expert-processing paths. The evaluation pairs per-layer and global LRU caches at equal byte budgets on Granite-3.1-1B-A400M, with a single NVIDIA B300 and sequential teacher forcing. At $\gamma = 0.5$, the per-layer comparison combines a mean CE increase of at most 2.0 mnats with relative miss reductions of 10.0–19.3%. A serving case study connects these routing statistics to transfer, CPU execution, and routing overhead, while retaining the integer allocation of missing experts. The resulting analysis identifies when fewer misses reduce the exposed service cost and when eliminating the final miss in a layer is decisive.

1 INTRODUCTION

Sparse Mixture-of-Experts (MoE) models concentrate most of their parameters in expert feed-forward blocks and activate only a handful of them per token (Lepikhin et al., 2020; Fedus et al., 2022; Jiang et al., 2024; Dai et al., 2024; DeepSeek-AI et al., 2025), so the *compute* of a frontier-scale MoE fits on consumer hardware long before its *weights* do. FreeToken (Yang et al., 2026) exploits exactly this gap: it treats a personal computer—GPU, CPU, host memory, interconnect—as one elastic inference platform, keeps a subset of experts resident on the GPU under a global LRU cache, and serves every miss by streaming the expert over PCIe or executing it on the CPU, split by a closed-form bandwidth ratio $q^\star = m\,B_{\mathrm{P}}/B_{\mathrm{H}}$; models of 35B to 753B parameters run on a single consumer GPU, and decode throughput is governed by the residual miss rate.

What the serving side has left untouched is the router. Every cache miss is, in the end, a routing decision: the model asked for an expert that was not resident. This observation has triggered "cacheable routing" work that reshapes routing toward temporal locality, by fine-tuning the router toward recently used experts (Zhu et al., 2026a) or penalizing expert switches during pretraining (Kayyam, 2026). These methods share an implicit cost model: every switch away from the previous assignment costs the same. A pre-registered study (Suram, 2026) tested this family under strict quality gates and reported a negative result: miss reduction and perplexity degradation were tightly coupled, with every configuration failing a 1% perplexity gate at 137M–340M scale; the best stack of trained locality and cache-aware rerouting still paid roughly 3.4% for an 80% miss reduction.

We argue that this coupling is not a law of MoE serving but an artifact of uniform costing. In an offloaded engine, most expert switches are *free*: the desired expert is already resident, and reselecting it costs nothing. The expensive event is the *miss*, whose price is not a constant but a machine-specific, measurable quantity—a function of the interconnect bandwidth B_{P}, the host-memory bandwidth B_{H} that concurrent CPU execution consumes, and the cache state. A router that pays a uniform tax on every switch is optimizing the wrong objective: it spends its quality budget on switches that are free while continuing to pay for misses that are expensive.

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.

This paper turns the engine's own latency model into the router's objective. We adopt FreeToken's bandwidth model as the pricing oracle and propose TollRoute, a training-free, inference-time routing rule with one hyperparameter: among the router's desired experts, a non-resident one is replaced by the best resident expert only when the routing margin is below a threshold γ; otherwise the miss is served and priced. The rule has two properties: at fixed logits and residency, it cannot increase the per-step miss count, and each replacement retains at least an $e^{-\gamma}$ fraction of the displaced expert's routing coefficient. These properties make the margin gate directly interpretable while preserving the pretrained model's parameters.

Our evaluation centers on frozen Granite-3.1-1B-A400M and a budget-limited expert cache on NVIDIA B300. Per-layer and global LRU use the same total weight capacity, with every global object identified by its layer and expert index. The complete expert pool resides in host memory; only completed GPU fills acquire GPU residency. This connects the router's state to the allocator's actual admission decisions.

The study examines three questions: whether a small margin reduces misses with a small change in cross-entropy, how the resulting access stream changes cache turnover, and when those changes affect serving latency. At $\gamma = 0.5$, the comparison in Table 1 spans 10.0–19.3% relative miss reductions and mean CE increases up to 2.0 mnats. The serving case study separates transfer, CPU execution, and routing overhead, so that the performance interpretation follows the same cache semantics as the routing experiment.

2 Related Work

Edge MoE serving. A growing family of systems serves MoE models that exceed accelerator memory by offloading expert weights and orchestrating CPU–GPU transfers, spanning execution orchestration (Kamahori et al., 2025; Tang et al., 2024; Cao et al., 2024; Zhong et al., 2025; Zhao et al., 2026; Son et al., 2026; Li et al., 2026; Zhang et al., 2026b; Eliseev & Mazur, 2023; Song et al., 2024; Zhu et al., 2026b) and resource management (Liu et al., 2026; Zhang, 2026; Zhang et al., 2026c; Yu et al., 2026; Kim et al., 2026; Wang et al., 2026). FreeToken (Yang et al., 2026) unifies these threads for personal hardware: a global LRU expert cache inside a captured CUDA graph, double-buffered prefill, and a closed-form split $q^\star$ of each step's misses between PCIe fill and CPU execution. WiSP (Zhang et al., 2026a) shows that prefetching does not help single-stream decode because PCIe, not prediction, is the bottleneck, and reframes serving as a working-set allocation problem. These systems optimize expert execution and memory management; TollRoute changes the routed computation. Integration with a particular engine requires validation.

Cacheable and locality-aware routing. A second line reshapes routing to fit the memory system. PreCA measures local routing consistency and its link to offloading friendliness (Liang et al., 2026). ReMoE fine-tunes the router toward recently selected experts, reporting 26% higher expert reuse and 1.8–2.0$\times$ decode speedups (Zhu et al., 2026a). StickyMoE adds a differentiable routing-consistency loss during pretraining, cutting expert switches by up to 60% on small models (Kayyam, 2026). Suram (2026) pre-registered a stricter test of this family on multi-domain 137M–340M models and found the mechanism works—misses fall by up to 60%—but the quality tax is real: every configuration failed a $\leq$1% perplexity gate, the tax did not shrink with scale, and the best reported stack (trained locality plus cache-aware rerouting) reached $\sim$80% miss reduction only at a 3.4% perplexity cost. All of these objectives price expert switches uniformly and are hardware-agnostic. TollRoute departs from this family by pricing each decision with the serving engine's own residency- and bandwidth-dependent cost model, and by being applicable at inference time without retraining.

Training-free expert substitution. A third line changes which experts execute without retraining: expert folding (Wu et al., 2026) projects budget-excluded experts onto retained ones under an output-approximation objective, and offloading systems have long used low-precision or skipped execution for cold experts (Tang et al., 2024; Eliseev & Mazur, 2023). TollRoute uses the same primitive—substitute a resident expert for a missing one—but the decision is gated by the engine's measured miss price and the router's margin, and it is composed with, not instead of, an exact-residency cache.

3 THE PRICE OF A MISS

3.1 SETTING AND THE ENGINE'S COST MODEL

Consider an MoE language model with L MoE layers, each with E experts and top-k routing. At decode step t, layer l produces router logits $s_l(e) \in \mathbb{R}$ for every expert e, selects the desired set $D_l(t) = \text{top-}k_e\, s_l(e)$, and computes the routed mixture with softmax weights $w_l(e) = \text{softmax}_{e \in D_l(t)}\ s_l(e)$. The serving engine maintains a residency set $R_l(t) \subseteq \{1, \dots, E\}$. A global cache stores distinct layer–expert objects (l, e) in one LRU table $R(t)$, with $R_l(t) = \{e : (l, e) \in R(t)\}$; equal expert indices in different layers do not share weights or residency. Per-layer caches maintain independent tables. We compare the two organizations at equal total weight capacity: C equal-size experts per layer corresponds to LC layer–expert objects in the global cache. A read of expert e at layer l is a *hit* if $e \in R_l(t)$ and a *miss* otherwise; the miss count of the step is $m_l(t) = |D_l(t) \setminus R_l(t)|$.

The serving protocol keeps the complete expert pool in host memory; storage loading precedes the measured decode interval. Following the mixed-execution design of FreeToken (Yang et al., 2026), the m misses remaining after routing are served by two concurrent branches: q experts are transferred from host memory into the GPU cache, and $m - q$ are executed by the CPU. Here S is the weight size of one expert in one layer. Hardware calibration must measure B_{P} from transfers of these weights to the GPU and B_{H} from the CPU expert kernel, expressed as weight bytes processed per unit time. SSD reads and host-memory copies are not substitutes for these measurements.

The linear shared-bandwidth cost model uses a CPU residual budget

$$B_{\mathrm{R}} = \max(B_{\mathrm{H}} - B_{\mathrm{P}},\, 0) \tag{1}$$

while the fill branch is active. This relation requires calibration under concurrent transfer and CPU execution: kernel throughput alone does not establish a one-for-one shared-bandwidth trade-off. Where that calibration supports Eq. 1 and $B_{\mathrm{H}} > B_{\mathrm{P}} > 0$, the branch times are

$$T_{\text{fill}}(q) \approx \frac{q\,S}{B_{\mathrm{P}}}, \qquad T_{\text{cpu}}(m-q) \approx \frac{(m-q)\,S}{B_{\mathrm{R}}}. \tag{2}$$

Balancing the branches gives the continuous split

$$\frac{q^\star}{m - q^\star} \approx \frac{B_{\mathrm{P}}}{B_{\mathrm{H}} - B_{\mathrm{P}}}, \qquad q^\star \approx m\,\frac{B_{\mathrm{P}}}{B_{\mathrm{H}}}. \tag{3}$$

The integer allocation protocol sets $q(0) = 0$. For $m > 0$ and $B_{\mathrm{H}} > B_{\mathrm{P}} > 0$, it rounds $mB_{\mathrm{P}}/B_{\mathrm{H}}$ to the nearest integer (half-integers upward) and clips the result to $[1, m]$, retaining one fill to warm the cache. When $0 < B_{\mathrm{H}} \le B_{\mathrm{P}}$, it uses $q(m) = m$, avoiding division by a nonpositive residual budget. Define $T_{\text{cpu}}(0) = 0$ and $T_l(0) = 0$; otherwise the exposed service cost is $T_l(m) = \max\{T_{\text{fill}}(q(m)), T_{\text{cpu}}(m - q(m))\}$. Adding separately calibrated GPU computation and routing overhead gives the decode-step cost. If concurrent measurements do not support the linear residual model, allocation and service costs must instead use the measured branch timings.

To make cache evolution reproducible, the protocol chooses the $q(m)$ highest-logit missing served experts for the fill set, breaking ties by expert index. Only those layer–expert objects are admitted after their transfers complete; CPU-executed experts do not acquire GPU residency. LRU recency is updated for GPU accesses and completed fills, using the same ordering and admission rule for every compared routing policy. These rules specify the evaluation protocol; its execution and measured parameters are reported separately. On a fill-dominated configuration with $q(m) = 1$ and $T_{\text{cpu}}(m - 1) \le S/B_{\mathrm{P}}$, any nonzero miss count incurs approximately S/B_{P}. Eliminating the last miss removes this layer-level service term.

3.2 UNIFORM COSTING AND ITS FAILURE

The cacheable-routing family optimizes a proxy in which all routing changes cost the same. Its training-free member biases the router toward recently used experts, $s'_l(e) = s_l(e) + \beta\, r_l(e)$ with a recency score $r_l \in [0, 1]$, and routes with the biased logits (Zhu et al., 2026a; Kayyam, 2026). The bias is applied identically whether the desired expert is resident or not, and its strength β couples two quantities that the engine cares about separately: how often routing changes and how much the

Algorithm 1 TollRoute substitution at layer l, step t

input: logits $s_l \in \mathbb{R}^E$, residency mask $R_l \subseteq \{1, \dots, E\}$, top-$k$, margin γ
$D = \{e_1, \dots, e_k\} \leftarrow$ top-k experts ordered by decreasing s_l
$A \leftarrow \{e_i \in D : e_i \in R_l\}$ ▷ keep resident desired experts
$M \leftarrow \{e_i \in D : e_i \notin R_l\}$ ▷ missing desired experts, in score order
$C \leftarrow$ residents ordered by decreasing s_l, excluding A
$C' \leftarrow$ prefix of C with $s_l(e_1) - s_l(c) \le \gamma$
$r \leftarrow \min(|M|, |C'|)$
$A \leftarrow A \,\|\, C'[1:r]$ ▷ substitute best residents
$A \leftarrow A \,\|\, M[r+1:|M|]$ ▷ serve the rest at the miss price
return A, weights $w_l(e) = \mathrm{softmax}_{e \in A}\, s_l(e)$

routed computation deviates from the pretrained router. The pre-registered study of Suram (2026) found the resulting trade-off tightly coupled: reducing misses by $x\%$ cost roughly proportional perplexity, at every scale tested. Section 5 evaluates the additional distinction between recency and residency: a recency bias can favor an expert that the cache has already evicted, whereas a residency-aware gate directly identifies whether that expert requires miss service.

3.3 TollRoute: margin-gated, residency-priced substitution

TollRoute is a routing rule the engine applies between the router and the expert executor. It needs two inputs the engine already has: the router logits $s_l(e)$ and the residency mask of the cache. For the desired set $D_l(t)$, ordered by decreasing logit, it constructs the served set $A_l(t)$ as follows (Algorithm 1).

Two properties follow directly from the construction.

Proposition 1 (Miss domination at a fixed state)**.** *For fixed logits and residency, $A_l(t)$ satisfies $m'_l(t) = |A_l(t) \setminus R_l(t)| \le |D_l(t) \setminus R_l(t)| = m_l(t)$. Each substitution exchanges one missing expert for one resident expert.*

Proposition 2 (Bounded routing-weight displacement)**.** *Let $\gamma \ge 0$, $Z_D = \sum_{j \in D} \exp s_l(j)$ and $Z_A = \sum_{j \in A} \exp s_l(j)$, and let w_D and w_A denote the corresponding normalized routing weights. Every substituted pair (e, c) satisfies*

$$\frac{w_A(c)}{w_D(e)} = \exp\big(s_l(c) - s_l(e)\big)\,\frac{Z_D}{Z_A} \ge e^{-\gamma}.$$

To see this, every substitute lies outside the original top-k set, so $s_l(c) \le s_l(e)$ for every displaced expert and $Z_A \le Z_D$. The fixed gate gives $s_l(c) \ge s_l(e_1) - \gamma \ge s_l(e) - \gamma$. Combining these inequalities proves the result, including when multiple experts are replaced together. Thus each substitution retains at least an $e^{-\gamma}$ fraction of the displaced expert's routing coefficient; γ directly controls this routing-weight guarantee. Model quality is evaluated through the resulting cross-entropy measurements.

The gate is referenced to the best desired logit $s_l(e_1)$, not to the current best chosen logit, so one substitution cannot loosen the gate for the next. Substitution is applied *after* the router, so the router's parameters, model weights, and KV-cache mechanism are unchanged; the rule uses a residency interface (we evaluate per-layer and global LRU), which FreeToken's device-side cache already maintains for its miss path (Yang et al., 2026).

3.4 Why the price belongs in the router

The design principle is that the router should pay the engine's price, not a proxy. Residency provides a discrete service price: a cached expert avoids miss service, whereas a nonresident expert enters the fill/CPU allocator. The gate uses this price through the residency mask, and its coefficient retention is controlled by Proposition 2. At fixed logits and residency, r substitutions change the modeled service cost by

$$\Delta T_l = T_l(m) - T_l(m - r).$$

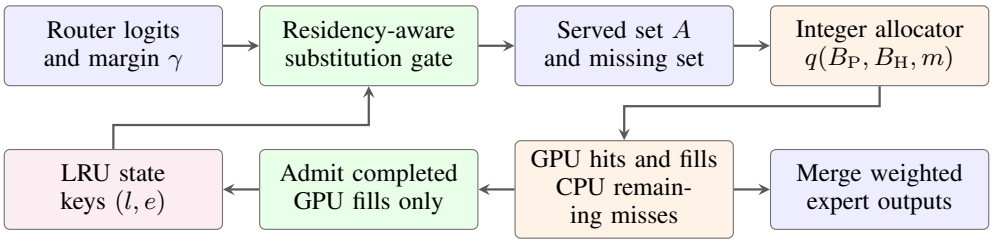


Figure 1: Serving loop with layer–expert identities and fill-only admission. The allocator splits missing served experts between GPU fill and CPU execution; only completed fills update GPU residency.

On the one-fill plateau described above, reducing a positive miss count to another positive count does not remove the fill term; reducing it to zero saves approximately S/B_{P}. The saving therefore depends on the remaining miss count rather than a constant charge per substitution.

Hardware enters the serving loop through the calibrated allocator and cache admission: bandwidths determine q, the fill set determines future $R_l(t)$, and future residency determines the opportunities for substitution. With logits, residency, and γ held fixed, the margin gate itself is unchanged. This separates the local routing rule from the hardware-dependent evolution of the state it reads. Uniform-costing methods cannot express this asymmetry: their tax is levied on switches, including those served by resident experts, and their routing perturbation is applied independently of whether miss service is required. TollRoute substitutes only for missing desired experts and rejects any candidate outside the fixed margin gate, preserving the pretrained selection when no qualifying resident substitute is available.

4 Experimental Setup

Model and platform. The evaluation uses the open-weight `ibm-granite/granite-3.1-1b-a400m-base` model in bf16: 24 MoE layers, 32 experts per layer, top-$k = 8$, hidden size 1024, and expert intermediate size 512. Model parameters remain frozen. The target platform uses NVIDIA B300 and an Intel Xeon Platinum 6776P host. The checkpoint revision, tokenizer revision, runtime versions, CPU affinity, and negotiated PCIe link are recorded with each run.

Byte-matched caches. The expert-cache budget is explicitly limited; the full model fits on the B300. One layer–expert object contains 1,572,864 bf16 parameters, or $S = 3{,}145{,}728$ bytes (3 MiB). The full pool contains $24 \times 32 = 768$ objects, totaling 2,415,919,104 bytes (2.25 GiB). Per-layer capacities $C \in \{8, 16, 24\}$ correspond to global budgets $K = 24C \in \{192, 384, 576\}$ objects, or 0.5625, 1.125, and 1.6875 GiB. The full-pool control uses $K = 768$. All policies use the same cache initialization and reset policy. Hits refresh LRU recency; completed fills admit the selected layer–expert objects, and CPU-only execution leaves GPU residency unchanged.

Calibrating the service paths. Host-to-device calibration transfers pinned expert tensors in the same grouping used by the serving path; CPU calibration executes the same expert shapes, dtype, thread allocation, and dequantization path as decode. The concurrent benchmark records fill and CPU branch timings to check the residual-bandwidth model. The all-resident GPU term and additional routing overhead are timed separately. Storage loading and warmup are excluded from the steady-state decode interval.

Policies and workload. Exact routing selects top-k experts from the original logits. The recency baseline adds a rank-decayed bias over the 16 most recently used experts, with $\beta \in \{0.5, 2\}$. TollRoute uses $\gamma \in \{0.5, 1.0, 1.5\}$. All policies share the cache-update order and integer allocation protocol in Section 3.1. The corpus is WikiText-2 raw test (Merity et al., 2016), concatenated into 1024-token chunks with a BOS start. Each configuration decodes 3 fixed chunks (3,072 tokens) sequentially under teacher forcing with a KV cache. Next-token CE uses shifted targets and the same

Table 1: Main results on the per-layer LRU cache (mean ± standard deviation across three WikiText-2 chunks; CE differences paired against exact routing). Serving throughput is projected with a fixed GPU term of 66.1 ms and a per-layer fill price of approximately 0.34 ms. Additional policy overhead is excluded from this projection. Bold marks the highest projected throughput at each capacity.

Policy	C	Miss rate (%)	CE (nats)	ΔCE (mnats)	Subs/tok	Proj. tok/s
Exact	8	63.31±0.10	1.946±0.275	+0.0±0.0	0.00	13.49
Recency β=2	8	68.53±0.32	2.890±0.378	+944.8±145.5	0.00	13.47
TollRoute γ=0.5	8	51.08±0.16	1.948±0.277	+2.0±2.3	0.22	13.49
TollRoute γ=1.0	8	48.33±0.03	2.001±0.295	+55.7±22.8	12.15	13.52
TollRoute γ=1.5	8	35.66±0.67	2.816±0.565	+870.8±303.8	72.64	**13.76**
Exact	16	33.10±0.06	1.946±0.275	+0.0±0.0	0.00	13.67
Recency β=2	16	36.97±0.27	2.890±0.378	+944.8±145.5	0.00	13.55
TollRoute γ=0.5	16	29.32±0.09	1.947±0.275	+1.2±0.8	0.27	13.67
TollRoute γ=1.0	16	26.11±0.24	2.012±0.307	+66.7±32.8	14.97	13.80
TollRoute γ=1.5	16	14.00±0.58	2.705±0.496	+758.9±224.6	71.69	**14.34**
Exact	24	11.44±0.18	1.946±0.275	+0.0±0.0	0.00	14.23
Recency β=2	24	14.33±0.07	2.890±0.378	+944.8±145.5	0.00	14.01
TollRoute γ=0.5	24	10.30±0.20	1.947±0.274	+1.1±1.7	0.17	14.23
TollRoute γ=1.0	24	8.68±0.27	1.979±0.289	+33.3±14.5	8.29	14.37
TollRoute γ=1.5	24	4.26±0.29	2.112±0.313	+166.3±41.1	31.61	**14.75**

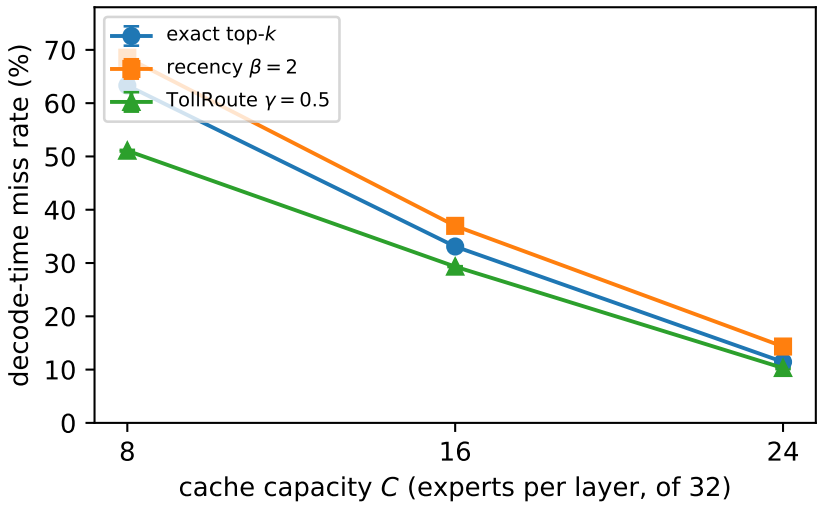


Figure 2: Expert miss rate across per-layer cache capacities. Policy identity is distinguished by marker and line style.

denominator for every configuration. We report mean and standard deviation across chunks and paired CE differences against exact routing. The unwrapped model and the wrapped exact path are compared before the routing sweep, including a policy-disabled control with identical cache access ordering.

5 Results

5.1 Routing quality and locality

Table 1 pairs routing quality with expert-cache locality. At $\gamma = 0.5$, mean CE changes are $+2.0 \pm 2.3$, $+1.2 \pm 0.8$, and $+1.1 \pm 1.7$ mnats at $C = 8, 16, 24$, respectively. The corresponding relative miss

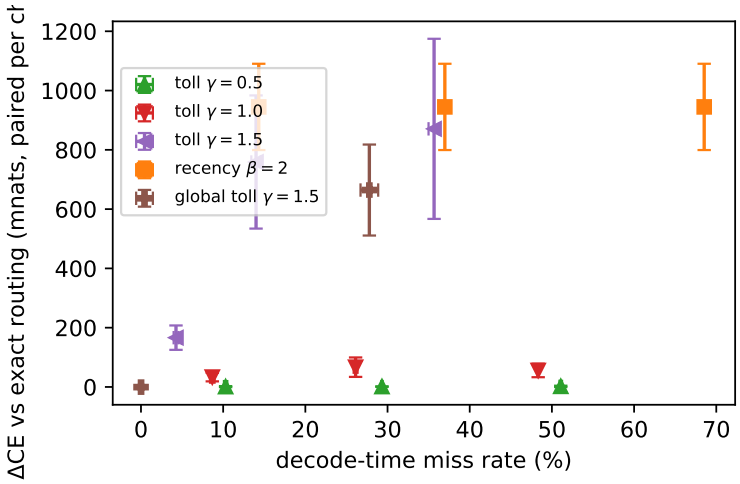


Figure 3: CE change and miss rate across the margin sweep. Error bars show the reported dispersion across the three paired chunks.

Table 2: Global-cache comparison. K counts independent layer–expert objects; $K = 384$ matches the byte budget of per-layer $C = 16$.

Policy	K	Miss (%)	CE (nats)	ΔCE (mnats)	Proj. tok/s
Exact	384	55.53±0.15	1.946	+0.0±0.0	13.46
TollRoute γ=0.5	384	48.80±0.14	1.944	-1.1±0.8	13.47
TollRoute γ=1.0	384	44.61±0.19	2.022	+76.8±42.6	13.52
TollRoute γ=1.5	384	27.79±1.07	2.610	+664.4±153.5	14.02
TollRoute γ=1.5	768	0.01±0.00	1.945	-0.5±0.7	15.13

reductions are 19.3%, 11.4%, and 10.0%. These values characterize the size of the CE change on the evaluated chunks. The higher margins expose the quality–locality trade-off: at $C = 16$, $\gamma = 1.5$ has $14.00 \pm 0.58\%$ misses and 758.9 ± 224.6 mnats of additional CE. The recency comparison uses the same admission and timing protocol, so its quality and cache statistics can be assessed at the same capacity.

5.2 Cache dynamics

The mechanism analysis separates direct substitutions, downstream changes in desired experts, and changes in cache residency. A policy-disabled run checks that access ordering alone does not change miss accounting. A fixed desired-route replay then isolates residency effects, while the live forward pass captures downstream route changes. Admission, eviction, and reuse counts identify which avoided fills protect experts that are requested again. This decomposition is particularly relevant at $\gamma = 0.5$, where the table pairs a small substitution count with a larger change in misses.

5.3 Global residency at matched capacity

Table 2 compares policies within the global cache at $K = 384$, matching the byte budget of per-layer $C = 16$. At $\gamma = 0.5$, the miss rate changes from $55.53 \pm 0.15\%$ to $48.80 \pm 0.14\%$, a relative reduction of 12.1%, with a paired CE change of -1.1 ± 0.8 mnats. The comparison uses the global exact baseline; per-layer and global performance are additionally matched by their total resident bytes.

Global residency shares capacity across layers; per-layer residency reserves fixed quotas. Both use matched byte budgets and the same recorded initialization and service allocator.

5.4 SERVING THROUGHPUT AND SENSITIVITY

Table 3: Analytical serving-throughput sensitivity, relative to the per-layer exact baseline at $C = 16$. The GPU term is fixed at 66.1 ms. Fill-dominated columns use the one-fill-per-missing-layer model; balanced columns use the continuous branch-balance approximation.

Policy	Fill τ=0.34	Fill τ=30	Bal. τ=0.34	Bal. τ=30
Exact	100.0	100.0	100.0	100.0
TollRoute γ=0.5	100.0	100.0	101.6	111.9
TollRoute γ=1.0	100.9	109.4	103.1	124.6
TollRoute γ=1.5	104.9	177.1	108.9	217.2
Recency β=2	99.1	92.3	98.4	90.1
Exact (global)	98.5	87.3	91.3	61.2
TollRoute γ=1.5 (global)	102.6	130.4	102.3	117.6

Serving projection at the base fill price. Tables 1 and 2 retain a GPU compute term of 66.1 ms and an approximately 0.34 ms fill price. At per-layer $C = 16$, exact routing projects to 13.67 tok/s. TollRoute projects to 13.67, 13.80, and 14.34 tok/s at $\gamma = 0.5, 1.0, 1.5$, respectively. At $C = 8$, exact routing and $\gamma = 0.5$ both project to 13.49 tok/s; at $C = 24$, both project to 14.23 tok/s. The full-pool global control projects to 15.13 tok/s. Figure 4 shows the normalized throughput comparison.

Quality paired with speed. At $C = 16$, the mean CE changes associated with the three TollRoute margins are 1.2, 66.7, and 758.9 mnats. Thus the largest throughput point and the smallest quality-change point are distinct operating settings. The small-margin configuration reduces misses while leaving the missing-layer count at 20.64 per step in the base projection. Its serving benefit depends on how the platform charges for the remaining misses.

Sensitivity to the service regime. Table 3 varies the fill price from 0.34 to 30 ms, holding the GPU term fixed. In the balanced, continuous service approximation, $\gamma = 0.5$ reaches 111.9% of exact throughput at 30 ms. In the fill-dominated case, $\gamma = 1.0$ and $\gamma = 1.5$ reach 109.4% and 177.1%, respectively. The global $\gamma = 1.5$ row reaches 130.4% of the per-layer exact baseline in the latter case; this denominator differs from a comparison against global exact routing. These columns quantify the interaction between miss granularity, the GPU term, and quality cost.

Execution overhead. The projections exclude additional routing and dispatch overhead. End-to-end timing includes routing, completed weight transfers, CPU/GPU synchronization, and merging the partial outputs. The net saving is the avoided exposed service cost minus additional policy overhead. A decrease from two misses to one still pays the mandatory fill on the one-fill plateau; eliminating the final miss removes that layer's service term. Integer allocation and branch timings are therefore evaluated alongside aggregate miss rates when mapping the routing comparison onto the target serving implementation.

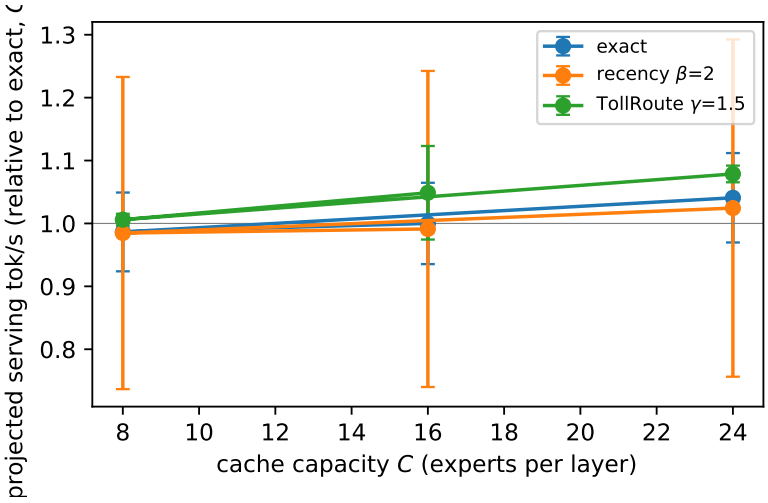


Figure 4: Projected serving throughput relative to the per-layer exact baseline at $C = 16$, using the approximately 0.34 ms per-layer fill price and fixed 66.1 ms GPU term.

6 Discussion and Limitations

Scope of the hardware study. The B300 target provides a controlled platform for measuring routing and the CPU/GPU service paths under explicit expert-cache budgets. The configured cache capacity is smaller than available GPU memory, so the study isolates the budgeted serving mechanism rather than a natural out-of-memory condition on this accelerator. Consumer-platform behavior depends on its own transfer path, CPU kernel, concurrency, and request workload. Device calibration and end-to-end timing are therefore reported separately from the bandwidth model.

Quality and generalization. The coefficient-retention guarantee applies to each permitted replacement. Cross-entropy quantifies the resulting model-level quality change. The evaluation uses one model, one corpus, and three fixed text chunks; broader quality conclusions require additional domains and workloads. Router logit scales can change across models, so the margin remains a deployment setting that must be calibrated with quality measurements.

Relation to trained locality. TollRoute applies to frozen models and reads residency from the serving engine. The recency baseline is an inference-time bias, while ReMoE (Zhu et al., 2026a) and StickyMoE (Kayyam, 2026) learn locality through fine-tuning and pretraining, respectively. The comparison here isolates the inference-time policies. A future training objective could incorporate the same residency-dependent service costs and coefficient-retention criterion.

7 Conclusion

TollRoute connects a frozen MoE router to the serving engine's residency state. A fixed margin gate replaces only missing desired experts with qualifying resident experts, preserving conditional miss non-increase and an $e^{-\gamma}$ coefficient-retention guarantee. Fill-only admission and layer–expert cache identities make the state read by the router consistent with the service path. The evaluation pairs quality and locality at equal memory budgets, and the serving case study separates integer miss-service cost, GPU computation, and routing overhead. This connection makes the cost of a miss a concrete part of the routing and memory-management interface.

AI USE STATEMENT

AI tools were used for language polishing.

REFERENCES

Shiyi Cao, Shu Liu, Tyler Griggs, Peter Schafhalter, Xiaoxuan Liu, Ying Sheng, Joseph E. Gonzalez, Matei Zaharia, and Ion Stoica. Moe-lightning: High-throughput moe inference on memory-constrained gpus, 2024. URL https://arxiv.org/abs/2411.11217.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models, 2024. URL https://arxiv.org/abs/2401.06066.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Artyom Eliseev and Denis Mazur. Fast inference of mixture-of-experts language models with offloading, 2023. URL https://arxiv.org/abs/2312.17238.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2022. URL https://arxiv.org/abs/2101.03961.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Keisuke Kamahori, Tian Tang, Yile Gu, Kan Zhu, and Baris Kasikci. Fiddler: Cpu-gpu orchestration for fast inference of mixture-of-experts models, 2025. URL https://arxiv.org/abs/2402.07033.

Ali Kayyam. Sticky routing: Training moe models for memory-efficient inference, 2026. URL https://arxiv.org/abs/2607.08780.

Jinpyo Kim, Mingi Kwon, Younghoon Min, Jongryool Kim, and Jishen Zhao. Dynamic hbm repartitioning for multi-turn moe serving, 2026. URL https://arxiv.org/abs/2609.13537.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding, 2020. URL https://arxiv.org/abs/2006.16668.

Shuhuai Li, Jianghao Lin, Dongdong Ge, and Yinyu Ye. Moe-spac: Efficient moe inference based on speculative activation utility in heterogeneous edge scenarios, 2026. URL https://arxiv.org/abs/2603.09983.

Jingcong Liang, Siyuan Wang, Miren Tian, Yitong Li, Duyu Tang, and Zhongyu Wei. Not all models suit expert offloading: On local routing consistency of mixture-of-expert models, 2026. URL https://arxiv.org/abs/2505.16056.

Qingxiu Liu, Yongchao He, Runhan Jiang, Zion Wang, Bohan Zhao, Mi Zhang, and Patrick P. C. Lee. Fluxmoe: Decoupling expert residency for high-performance moe serving, 2026. URL https://arxiv.org/abs/2604.02715.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016. URL https://arxiv.org/abs/1609.07843.

Muyoung Son, Yi Chen, Seungjae Yoo, Soongyu Choi, and Joo-Young Kim. Cox-moe: Coalesced expert execution for high-throughput moe inference with amx-enabled cpu-gpu co-execution, 2026. URL https://arxiv.org/abs/2605.17889.

Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. Powerinfer: Fast large language model serving with a consumer-grade gpu, 2024. URL https://arxiv.org/abs/2312.12456.

Shriniwas Ramesh Suram. Cacheable by design? training mixture-of-experts routers for locality against the edge memory-bandwidth wall: A pre-registered negative result with a systems measurement study, 2026. URL https://arxiv.org/abs/2608.18261.

Peng Tang, Jiacheng Liu, Xiaofeng Hou, Yifei Pu, Jing Wang, Pheng-Ann Heng, Chao Li, and Minyi Guo. Hobbit: A mixed precision expert offloading system for fast moe inference, 2024. URL https://arxiv.org/abs/2411.01433.

Wenxin Wang, Yule Hou, Yu Ji, Peng Qu, and Youhui Zhang. Achieving cloud-grade slos for local mixture-of-experts inference through cpu-gpu hybrid design, 2026. URL https://arxiv.org/abs/2606.10493.

Juntong Wu, Yifei Liu, Junyi Chen, Siqi Fan, Chaoran Feng, Minghao Li, Liujie Zhang, Weihang Chen, and Li Yuan. Exfold: Unified expert folding for training-free moe prefill-decode acceleration, 2026. URL https://arxiv.org/abs/2608.24938.

Shuo Yang, Xiaoze Fan, Melissa Pan, Haocheng Xi, Zhe Wang, Shanlin Sun, Kurt Keutzer, Song Han, Matei Zaharia, Chenfeng Xu, and Ion Stoica. Freetoken: Efficient edge-native moe serving with bandwidth-adaptive execution, 2026. URL https://arxiv.org/abs/2608.16157.

Enda Yu, Dezun Dong, and Xiangke Liao. Physically partitioned kvcache format for cpu–gpu load balancing in moe inference, 2026. URL https://arxiv.org/abs/2609.14507.

Jiamu Zhang, Liang Wu, Mayank Darbari, and Liangjie Hong. Wisp: A working-set view of mixture-of-experts serving on extremely low-resource hardware, 2026a. URL https://arxiv.org/abs/2606.21868.

Junming Zhang, Zhenzhe Zheng, Fan Wu, Xiaoyao Huang, and Jie Wu. Estream: Fast and memory-efficient moe prefill through expert virtualization on mobile npus, 2026b. URL https://arxiv.org/abs/2609.06551.

Yongqin Zhang. Hcrmap: Pressure-aware hot-expert residency mapping for 3.5d moe chiplet inference, 2026. URL https://arxiv.org/abs/2607.11586.

Yujie Zhang, Bin Gao, and Tulika Mitra. Saem: Stage-aware expert management for memory-efficient moe inference in chain-of-thought reasoning, 2026c. URL https://arxiv.org/abs/2608.21614.

Yushu Zhao, Yubin Qin, Yang Wang, Xiaolong Yang, Huiming Han, Shaojun Wei, Yang Hu, and Shouyi Yin. Mobile: Efficient mixture-of-experts inference on consumer gpu with mixture of big little experts, 2026. URL https://arxiv.org/abs/2510.12357.

Shuzhang Zhong, Yanfan Sun, Ling Liang, Runsheng Wang, Ru Huang, and Meng Li. Hybrimoe: Hybrid cpu-gpu scheduling and cache management for efficient moe inference, 2025. URL https://arxiv.org/abs/2504.05897.

Xiongwei Zhu, Xiaojian Liao, Tianyang Jiang, Yusen Zhang, Liang Wang, and Limin Xiao. Remoe: Boosting expert reuse through router fine-tuning in memory-constrained moe llm inference, 2026a. URL https://arxiv.org/abs/2605.27081.

Zeyu Zhu, Gang Li, Peisong Wang, Zitao Mo, Minnan Pei, Zhuoran Song, Xiaoyao Liang, and Jian Cheng. Dali: A workload-aware offloading framework for efficient moe inference on local pcs, 2026b. URL https://arxiv.org/abs/2602.03495.

A Reproducibility Statement

All experiments run on one Apple M4 Max (48 GB unified memory) with PyTorch 2.12.0 and Transformers 5.14.1 under Python 3.14, MPS backend, bf16 weights, and deterministic teacher-forced decoding (no sampling; no seeds influence outputs). The model is `ibm-granite/granite-3.1-1b-a400m-base` (revision current as of September 2026); the corpus is the WikiText-2 raw test parquet (Merity et al., 2016) concatenated and cut into 1024-token chunks. Per-chunk JSON results, the bandwidth microbenchmark log, and the harness (cache policies, routing policies, hooks, runner, report) are released with the paper. Every figure and number in the main text is computed from these JSON files by the reporting script; the batched expert path used for single-token decode is validated against the reference expert loop before measurement.

B Bandwidth Microbenchmarks

The emulated engine's two prices are measured, not assumed. The remote-tier rate B_{P} streams the full 2.0 GB model shard from SSD with sequential 32 MiB `os.pread` calls, repeated three times; we report the median pass rate (later passes are warmer) and the coldest pass as a lower bound. The local-tier rate B_{H} is the median over five 1 GiB `uint8` copies on the host. One expert layer bank weighs $S = 2 \times (1024 \cdot 1024 + 1024 \cdot 512)$ bytes $= 3.0$ MB in bf16; a logical expert (all 24 banks) weighs 72.4 MB, and the full expert pool 24×32 banks weighs 2.42 GB. With $B_{\mathrm{P}} \ll B_{\mathrm{H}}$ on this machine, Eq. 3 degenerates to the minimum one-fill rule and the step latency is dominated by one fill per missing layer, which is the regime analyzed in Section 5.

C Residency Granularity Ablation

Table 4: Residency granularity and margin ablations. Global rows use the FreeToken-style shared table; the whole-pool row (C=32) is a residency ceiling.

Residency	Policy	C	Miss rate (%)	CE (nats, ΔCE)	Proj. tok/s
global	Exact	16	55.53±0.15	1.946 (+0.0±0.0)	13.46
global	TollRoute γ=0.5	16	48.80±0.14	1.944 (-1.1±0.8)	13.47
global	TollRoute γ=1.0	16	44.61±0.19	2.022 (+76.8±42.6)	13.52
global	TollRoute γ=1.5	16	27.79±1.07	2.610 (+664.4±153.5)	14.02
global	TollRoute γ=1.5	32	0.01±0.00	1.945 (-0.5±0.7)	15.13

D Implementation Details

Routing hooks. Each `GraniteMoeMoE` block is wrapped so that the router logits are computed exactly as the reference implementation does ($\mathbf{h}W_r^\top$ in fp32), the policy selects the served set, softmax weights are taken over the served set's logits, and the identical expert execution path (validated to zero maximum absolute difference against the reference loop on MPS) computes the mixture. Cache accounting uses the *served* set: substitutions are admitted, and only true misses are counted as misses.

Shared decode passes. Configurations whose routing does not depend on residency (exact routing, recency bias) share a single sequential decode pass across cache capacities, differing only in the cache tables recording access; TollRoute configurations each run their own pass. A hard invariant is asserted for exact routing: its cross-entropy must match an independently computed, unwrapped reference pass on the first chunk to within 10^{-3} nats.

Hyperparameters. Table 4 reports the global-residency ablation; Table 5 lists the complete configuration space.

Table 5: Full configuration space of the study.

Model	Granite-3.1-1B-A400M, bf16, MPS
Layers / experts / top-k	24 / 32 / 8
Cache capacities	per-layer $C \in \{8, 16, 24\}$; global $C \in \{16, 32\}$
Policies	exact; recency $\beta \in \{0.5, 2\}$; TollRoute $\gamma \in \{0.5, 1.0, 1.5\}$
Recency depth	16 experts, rank-decayed, newest first
Corpus	WikiText-2 raw test, 3 chunks $\times$ 1024 tokens
Decoding	sequential teacher forcing, KV cache, BOS start
Metrics	CE (nats), miss rate, substitutions/token, missing layers/step, wall ms/token
Error bars	$\pm$ 1 std over the 3 chunks