Hydra: Exactly Isolated Tenant Lanes on One Shared Language Model

Throughput, memory footprint and byte-exact isolation on Qwen3-8B and Gemma 3 4B and 12B

Ryan R

Sekos AI

September 2026

Abstract

Serving one language model to many tenants forces a choice: deploy a model per tenant and pay for every copy, or share one model and trust that nothing leaks between tenants. Hydra removes that choice. Each tenant owns a *lane*: private attention (as a low-rank delta over the shared projections) and private normalization gains, over one frozen, shared backbone. Authority is signed to a lane and its full transformer head: no request reaches a lane without a single-use capability bound to it, and every kernel that touches a lane's row has a shape fixed by that lane's own history and deployment constants, never by its neighbors. The result is exact rather than approximate: a lane's logits are byte-identical whatever its neighbors' prompts, weights, arrival order or timing, and equal the plain model running that lane alone.

Shared-cache serving engines are not an option for isolated tenants, so we compare against the isolated alternative: the same model serving tenants one at a time. On an Apple M5 Max, three lanes on one Qwen3-8B serve 1.85× that throughput on a 72-instance coding workload (34.7 vs 18.8 tokens/s), and up to 2.58× at row capacity 16, with every lane token-identical to its reference on 72 of 72 instances in every repetition. Gemma 3 4B and 12B, added in this work with sliding-window attention, reach 2.24× and 2.43×. A tenant's private state is 29.8 MiB at 8B (0.19% of the model), so sixteen isolated tenants need 15.7 GiB of weights instead of 244 GiB for sixteen model copies. Serial acceptance runs (H3) are byte-exact on every lane of Gemma 3 4B, Qwen3-8B and Gemma 3 12B, and a 20-case authenticated HTTP roster (H2) passes on Gemma 3 4B and 12B. Across every evaluation, including earlier designs measured on NVIDIA L4, A100 and H200 GPUs and on CPU with Qwen2.5-0.5B, Qwen3-0.6B and 4B and Whisper, the exactness ledger compares 6.1 billion logits and activations and 930,337 generated tokens with their references: every one is byte-identical. We report the timing side channel that remains outside this guarantee, the defects our own evaluation found, and what is not yet qualified.

1 Introduction

1.1 What a Hydra is

A Hydra is one language model serving many tenants, each with its own *head*, a private lane, on a shared body. The body is the model's frozen weights: embeddings, feed-forward layers and output head, loaded once. A lane is everything that makes a tenant's inference its own: private attention (a small low-rank correction to each shared attention projection), private normalization, and the key-value cache of every request the tenant sends. Lanes are small enough that one machine holds many, and their arithmetic is fixed so that nothing another tenant sends, holds or does can change a single bit of a lane's output.

Every lane sits behind the Schemen Gate[1], the open-source authority layer of AI PKI and enforcement primitives. Authority is signed to a lane and its full transformer head, not to a tenant: a request reaches a lane only through a single-use capability bound to that lane (with the request's body digest, action, model, runtime, expiry and a one-time identifier). Before a lane opens, the Gate sidecar, a separate process holding key material the runtime never sees, must issue that lane's Cryptographic Dimension Partitioning mask [1]; the sidecar also verifies each lane's signed artifacts at admission. Only then does the runtime open the lane's route; every forward re-checks each row's lease and each lane's weight custody, and every result carries a custody receipt. A missing, replayed, expired or misbound capability is refused before any private lane is touched, and losing the sidecar stops generation.

CDP gives each tenant an exact, disjoint share of the feed-forward coordinates but leaves attention shared, since every query meets every key. Hydra supplies the missing piece: a duplicated attention lane per tenant.

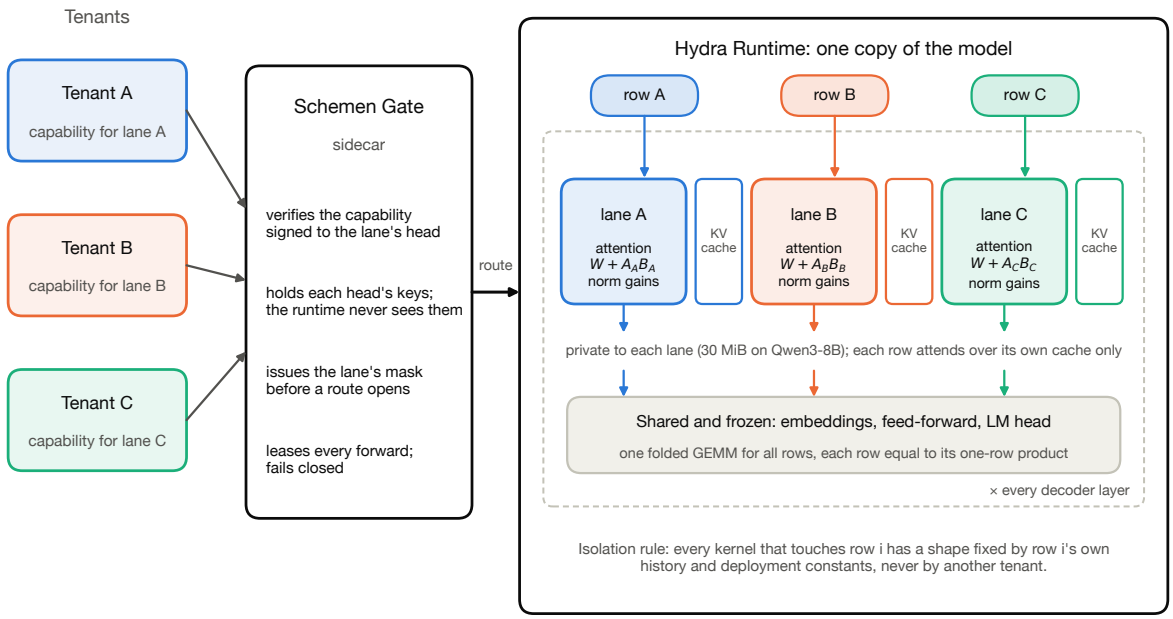

Figure 1: A Hydra. Authority is signed to a lane and its full transformer head; a request reaches a lane only through a capability bound to it. The Gate sidecar verifies it, holds the lane keys the runtime never sees, issues the lane's mask and leases every forward. In the runtime, every row passes through its own lane's private attention and norms over its own key-value cache, then through the one shared, frozen copy of the rest of the model, which serves all rows in one folded GEMM.

1.2 Why build one

A law firm's language model should not let one client matter inform another's answers. The same holds for a hospital's departments, a bank's customers or an agent platform's users. Today this is usually handled one of two ways. Either each tenant gets its own deployment, which multiplies weight memory and leaves each copy mostly idle, or tenants share one deployment and isolation rests on application logic around a model whose batching, caching and kernel choices were never designed to keep tenants apart.

Hydra is a third option. It serves many tenants from one copy of the weights while making isolation a property of the arithmetic. A tenant's *lane* owns its attention (a rank-k correction to each shared projection) and all of its normalization gains; embeddings, feed-forward layers

[1] https://github.com/sekosai/schemen-gate

and the output head are shared and frozen. Requests carry signed capabilities that bind the principal, the lane, the operation and the exact request body; every forward re-checks each row's authority and the custody of every lane's parameters, and fails closed.

Batching tenants together is where serving systems usually give up exactness: GEMM libraries switch kernels with the row count, padding changes attention shapes, and a neighbor's long prompt changes the chunking of yours [7]. Hydra's *isolation rule* is that every kernel touching row i has a shape fixed only by row i's own history and deployment constants. With that rule a lane's logits are byte-identical whatever its neighbors do, which we test directly by comparing SHA-256 digests of logits and exact token sequences.

Contributions.

- An exact multi-tenant serving design (Section 3): private attention and norms per lane over a shared backbone, low-rank delta lanes in the Punica/S-LoRA structure [3, 4], device-qualified folded decode GEMMs, per-row attention over each row's own cache, per-lane chunked prefill and continuous (iteration-level) scheduling [5], all under per-request capability authority.
- An extension to Gemma 3 [8]: sliding-window attention with stock cache and mask geometry per row, sandwich norms and final logit softcapping, in every execution mode.
- Measurements on one Apple M5 Max (Section 6, 7): throughput, memory footprint and latency on Qwen3-8B [9] and Gemma 3 4B and 12B, with exactness checked on every instance of every run.
- Exactness evidence (Section 5): an exactness ledger of 6.1 billion values and 930,337 tokens, all byte-identical; and byte-exact serial acceptance (H3) on Gemma 3 4B, Qwen3-8B and Gemma 3 12B, including interventions that change one lane and leave every other lane byte-identical.
- Authorization evidence (Section 9): a 20-case authenticated HTTP roster (H2).
- An account of what is outside the guarantee (timing), the defects the evaluation itself exposed, and the remaining qualification work.

2 Background

Cryptographic dimension partitioning. The companion CDP paper [1] binds an authenticated tenant scope to gate masks that partition feed-forward coordinates exactly: masks are disjoint and exhaustive, so an excluded coordinate is exactly zero. CDP states that it does not partition shared attention, and that moving attention outside the shared boundary would require "duplicating complete Transformer lanes, including normalization and every residual route". Hydra is that construction, with an exactness argument for the batched execution CDP does not cover.

Multi-tenant low-rank serving. Punica [3] and S-LoRA [4] run one base GEMM over a batch of requests and add each request's LoRA [2] correction with a gathered batched kernel. Hydra's delta lanes use that structure, with one addition: each kernel's shape is fixed by the roster and deployment constants so that no lane can change another lane's bits.

Batch invariance. Kernel results depend on batch shape because libraries pick different reduction orders for different shapes [7]; serving engines have since added deterministic modes, built for reproducibility rather than for keeping tenants apart.

Why shared-cache serving is not a baseline here. General serving engines such as vLLM [10] and SGLang [11] earn much of their throughput by sharing state across requests: paged key-value caches, automatic prefix reuse [11, 6] and semantic caches. For tenants that must be isolated this is disqualifying, not merely weaker. Shared prefix, key-value and semantic caches leak other users' prompts through timing [12, 13]; mitigations restrict sharing rather than remove the channel [14]; and batch-dependent arithmetic lets one request change another's output. None binds a per-request capability to the exact lane and body it may touch. The admissible alternatives for isolated tenants are therefore a deployment per tenant, or one model serving tenants one at a time, and that is the baseline in every comparison here. Hydra's lanes have private key and value projections, so there is no prefix or key-value state to share across tenants in the first place.

3 Design

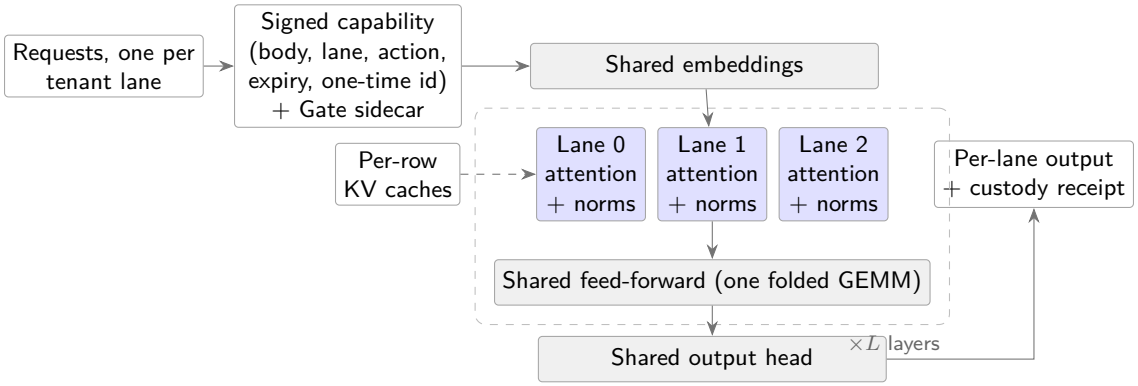

Figure 2: Hydra serving several tenants from one copy of the weights. Each lane owns its attention (a low-rank delta over the shared projections) and its normalization gains; each request row owns its key-value cache. Everything else is shared and frozen. Every forward re-checks each row's authority and each lane's custody.

3.1 Lanes and the shared backbone

A Hydra deployment holds one backbone and R lanes. Every decoder attention module and every normalization site becomes a *bank* with one member per lane. A lane either aliases the backbone's operators (a base clone), owns dense private copies, or owns rank-k factors $A_\ell \in \mathbb{R}^{k\times d_{\text{in}}}$, $B_\ell \in \mathbb{R}^{d_{\text{out}}\times k}$ per projection, computing $xW + (xA_\ell^\top)B_\ell^\top$ with the backbone's W shared. Base weights are never accepted in a delta state, so a lane can neither replace nor copy the shared operator. The feed-forward layers, embeddings and output head are shared and frozen.

3.2 Authority

Every HTTP request carries an Ed25519 capability over its canonical body that binds subject, action, lane set, model, runtime, expiry and a one-time identifier (replays are refused from a persistent store). Masks come from a separate Gate sidecar process over an owner-only socket; losing the sidecar fails closed. Every banked forward re-checks each row's lease and a custody fingerprint of every lane's parameters; results carry a custody receipt that binds what actually ran. Lane artifacts are admitted only through signed manifests at cold start, and a byte-tampered artifact is refused before serving.

3.3 The isolation rule, and how grouped execution keeps it

Folded decode GEMMs. One GEMV per decode row re-reads the weights once per row; folding rows into one GEMM reads them once. GEMM libraries change reduction order with the row count, so the fold width is *qualified* at load: on the deployed device and weights, the widest fold whose every row equals the single-row product byte for byte, over eight heavy-tailed probes (*identity fold*; 4 rows on MPS bf16). An opt-in *fixed fold* runs one GEMM as wide as the row capacity; it is verified at load to be neighbor- and position-invariant and fails closed otherwise, and a lane then equals the plain model run on the same fold.

Per-row attention. Attention runs per row over that row's own cache, so a row's attention shape depends on its history alone; decode folds each key-value group's query heads onto the query axis instead of expanding keys and values.

Per-lane prefill. Each request is prefilled alone, in chunks on its own grid; neighbors never change where its prompt is split.

Continuous scheduling. Requests join and leave between decode steps with a fixed row capacity, so joins and leaves never change any request's arithmetic.

3.4 Gemma 3

Gemma 3 interleaves five sliding-window (local, 1,024 tokens) layers with one global layer, uses four normalization sites per layer with a $(1 + \gamma)$ gain in fp32, and softcaps final logits in some configurations [8]. Each request's cache uses Transformers' own sliding-window layers (a bounded variant adds only an admitted token aperture), and each row's masks are built from that row's cache by Transformers' own mask builders, exactly as the plain model builds them. Custody seals each cache layer's type and window, and every step checks that the cache layout matches the model's layer types; a full-attention layer standing in for a sliding one would silently widen what a row sees. All four sandwich norms and the q/k norms are private per lane. The released 4B and 12B checkpoints are multimodal; we serve their exact text decoder, which matches the multimodal model's text path byte for byte.

4 Experimental setup

All measurements are on one Apple M5 Max (128 GB unified memory, MPS), PyTorch 2.12.1, Transformers 5.9.0, bf16. Models: Qwen3-8B (snapshot `b968826d`), Gemma 3 4B (`093f9f38`) and 12B (`96b6f1ec`), instruction-tuned.

Workload. A deterministic coding suite of 72 instances over a small library written for the evaluation (119 tests): 24 bug fixes, 24 function implementations and 24 execution-prediction questions, each at a file tier (about 1k prompt tokens) and a repository tier (about 7.5k), 308,200 prompt tokens in total. Answers are graded by running the library's tests. All 72 requests arrive at once; greedy decoding, at most 1,024 new tokens, prefill in 1,024-token chunks in every arm.

Arms. The *plain model* is the unmodified Hugging Face model serving one request at a time. Hydra runs three lanes under the continuous scheduler at row capacity 3, 6 and 16 (the last with the fixed fold). Throughput runs use *zero lanes*: rank-16 deltas whose B is exactly zero, so every delta kernel runs while each lane is mathematically the plain model; this makes token identity to the plain model the test of isolation. A separate arm uses private lanes (random rank-16 deltas at 2% of each projection's norm, one seed per lane).

Validity. Each run is a fresh process. A monitor samples the process table every 5 s; if another model-sized process runs, the run is voided and repeated. Voided runs are kept and labeled, never silently dropped, and never enter any statistic here. Throughput is generated tokens over engine wall time; intervals are t-based 95% confidence intervals across repetitions. Time to first token (TTFT) p50 is the mean of per-repetition medians. Footprint is the kernel-charged process footprint (macOS physical footprint, Metal buffers included), its high-water mark over the serving phase. The machine is shared, so host CPU load from other processes is recorded for every run rather than voided: MPS decode dispatch is partly host-side, so such load can slow a run but not speed it up. 60 of the 83 kept runs, plain-model and Hydra alike, shared the host with other processes using a median of 207–511% of a core; the per-run figures are in the released run table.

5 Exactness

Hydra's isolation is byte-exact, not approximate, and no comparison in this paper uses a tolerance. Table 1 lists every exactness comparison behind it: 6.1 billion logits and activations and 930,337 generated tokens, compared across 9 kinds of evaluation, 8 models and 5 kinds of device. Every one is byte-identical to its reference: 0 mismatches. On the coding workload alone, 58 zero-lane Hydra runs across every model, row capacity, fold and runtime build served 4,176 requests, and all 927,169 generated tokens equal the plain model's.

Evaluation	Model	Device	Comparisons	Values	Tokens	Mismatches
Coding workload, served	Qwen3-8B	M5 Max	3,096	–	589,140	0
Coding workload, served	Gemma 3 12B	M5 Max	432	–	164,836	0
Coding workload, served	Gemma 3 4B	M5 Max	648	–	173,193	0
Serial acceptance (H3)	Qwen3-8B	M5 Max	88	13.4 million	1,024	0
Serial acceptance (H3)	Gemma 3 4B	M5 Max	216	56.6 million	1,024	0
Serial acceptance (H3)	Gemma 3 12B	M5 Max	216	56.6 million	1,024	0
Prototype bank replication, R=2	Qwen3-4B-Instruct	A100	128	1.2 billion	–	0
Prototype bank replication, R=4	Qwen3-4B-Instruct	A100	256	2.4 billion	–	0
Prototype bank equivalence, R=1, 2, 4	Qwen2.5-0.5B-Instruct	L4	1,024	1.9 billion	–	0
Prototype lifecycle, R=8	Qwen3-4B-Instruct	A100	448	68.1 million	–	0
Prototype lifecycle, R=16	Qwen3-4B-Instruct	A100	896	136.1 million	–	0
Prototype lifecycle, R=32	Qwen3-4B-Instruct	H200	1,792	272.3 million	–	0
Prototype trained-lane interventions	Qwen3-4B-Instruct	H200	1,792	–	–	0
Runtime serial acceptance, fp32	Qwen3-0.6B	CPU	4	–	64	0
Runtime serial acceptance, fp32	Qwen3-4B	CPU	4	–	32	0
Whisper transcripts, R=1 and R=4	Whisper tiny.en	CPU	1,000	–	–	0
Whisper foreign interventions	Whisper tiny.en	CPU	864	–	–	0
Total	8 models	5 devices	12,904	**6.1 billion**	**930,337**	**0**

Table 1: The exactness ledger: every comparison behind this paper, read from its source record. "Values" counts logit and activation scalars where the compared shape is recorded (full-vocabulary vectors); "Tokens" counts generated tokens compared. Prototype rows are the research implementation that preceded the Runtime (Section 10).

5.1 Each lane is its own model

Model	Lanes	Logits exact	Greedy tokens identical	Sliding window (cached decode)	Foreign intervention	Lifecycle	Shared weights sealed
Gemma 3 4B	4	4/4	4/4 (256 steps)	4/4 (1,120 vs 1,024)	pass	pass	–
Qwen3-8B	4	4/4	4/4 (256 steps)	n/a (no sliding layers)	pass	pass	yes
Gemma 3 12B	4	4/4	4/4 (256 steps)	4/4 (1,120 vs 1,024)	pass	pass	yes

Table 2: Serial acceptance (H3) on MPS in bf16: every lane against an independent native reference built by grafting that lane's operators into a fresh plain model. Logits are compared with `torch.equal` at a fixed shape; greedy tokens over 256 steps; the sliding-window case runs a prompt longer than the window through cached generation. The foreign intervention changes one lane's weights and requires every other lane to stay byte-identical; the seal proves no path wrote to the shared backbone.

Table 2 is the direct test of the claim: each lane equals the plain model running that lane's operators alone, byte for byte, for every model.

5.2 No influence across lanes

Equality with a reference shows a lane computes its own model; isolation also needs the converse, that nothing a neighbor is or does reaches it. Serial acceptance changes one lane's weights and requires every other lane to stay byte-identical (Table 2). A crossover probe on Qwen3-8B with three *different* private lanes found every request's logits unchanged at every step under reversed arrival and when one lane's private weights were replaced, while the changed lane itself changed. On the research prototype, 256 trained-lane donor interventions left all 1,792 receiver comparisons exact, and Whisper's 288 interventions moved none of 864 foreign comparisons (Table 1).

Defects the evaluation found. Exactness testing at full scale found four defects that unit tests had not, each now fixed with a regression test:

1. A shared prefill budget split a request's prompt at a boundary set by its neighbors (70/72 identical); prefill now chunks on each request's own grid.
2. A fold width qualified with a single probe passed a kernel that differed at 5–6 rows (50/72); qualification now uses eight heavy-tailed probes.
3. The opt-in fixed fold checked neighbor values but not row position; CPU fp32 GEMMs round by position, so arrival order could move a lane's bits. The fold now verifies position invariance and fails closed.
4. Hydra's native loader rounded RoPE frequency buffers to bf16, so served lanes drifted from the stock model (isolation was unaffected, since all lanes shared the error). Frequencies are rebuilt in fp32. The serial acceptance harness had the same rounding and had granted bf16 a tolerance that hid it; it now requires byte-exactness at every dtype.

6 Throughput

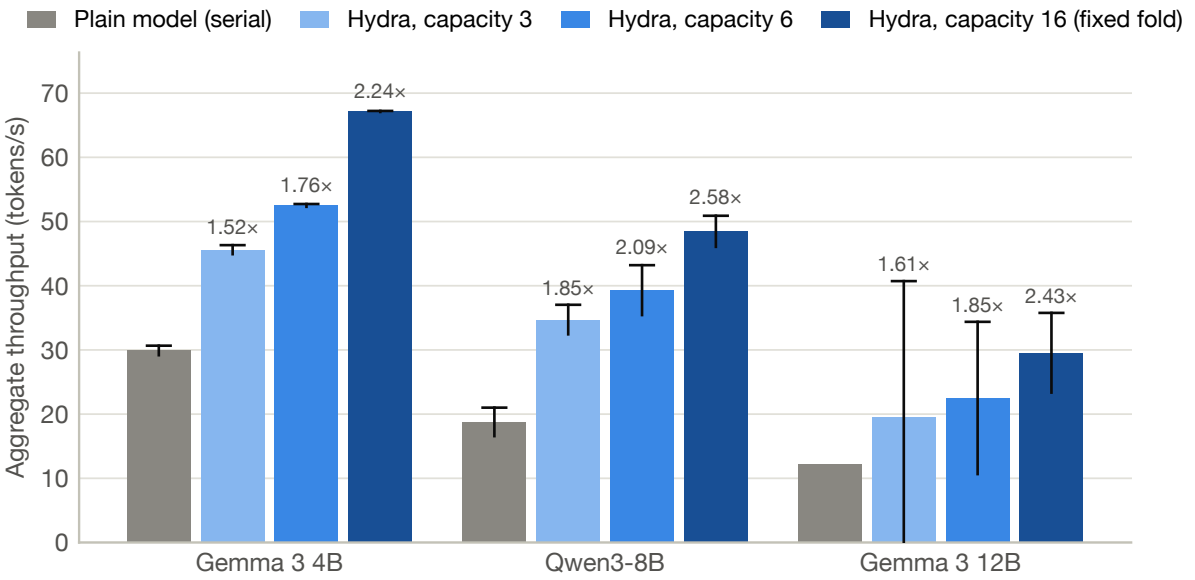

Figure 3: Aggregate throughput on the 72-instance coding workload: the plain model serving one request at a time against three Hydra lanes on one copy of the same model at row capacity 3, 6 and 16 (fixed fold). Whiskers are 95% confidence intervals across repetitions; labels give the ratio to the plain model. Every Hydra run is token-identical to its reference on 72/72.

Model	Configuration	Tokens/s (95% CI)	× plain	Identical to reference	Pass /72	Footprint HWM (GiB)	TTFT repo p50 (s)	TPOT p50 (ms)
Gemma 3 4B	Plain model, serial (n=3)	29.9 (29.2–30.7)	1.00	–	5	19.1	2.45	30
	Plain model, fold 16 (n=1)	25.5	0.85	33/72	6	19.4	2.60	34
	Hydra, capacity 3 (n=3)	45.6 (44.9–46.3)	1.52	72/72	5	17.0	2.84	52
	Hydra, capacity 6 (n=3)	52.5 (52.3–52.7)	1.76	72/72	5	18.9	3.05	102
	Hydra, capacity 16, fixed fold (n=3)	67.2 (67.1–67.2)	2.24	72/72	6	24.7	3.59	214
	Hydra, capacity 6, private deltas (n=1)	53.1	1.77	n/a	6	18.7	3.07	101
Qwen3-8B	Plain model, serial (n=3)	18.8 (16.6–21.0)	1.00	–	36	24.6	3.88	43
	Plain model, fold 16 (n=3)	16.6 (16.3–16.9)	0.88	44/72	39	24.4	4.45	48
	Hydra, capacity 3 (n=3)	34.7 (32.4–37.0)	1.85	72/72	36	27.7	4.32	71
	Hydra, capacity 6 (n=3)	39.3 (35.4–43.2)	2.09	72/72	36	32.2	4.48	136
	Hydra, capacity 16, fixed fold (n=3)	48.5 (46.1–50.9)	2.58	72/72	39	44.8	5.92	278
	Hydra, capacity 6, private deltas (n=1)	40.3	2.14	n/a	37	31.9	4.35	134
Gemma 3 12B	Plain model, serial (n=1)	12.2	1.00	–	30	55.9	7.44	73
	Plain model, fold 16 (n=1)	10.8	0.88	41/72	28	55.3	8.49	82
	Hydra, capacity 3 (n=2)	19.5 (-1.6–40.7)	1.61	72/72	30	38.4	9.03	129
	Hydra, capacity 6 (n=2)	22.5 (10.7–34.4)	1.85	72/72	30	41.6	9.34	245
	Hydra, capacity 16, fixed fold (n=2)	29.6 (23.4–35.8)	2.43	72/72	28	53.7	10.49	489
	Hydra, capacity 6, private deltas (n=1)	22.3	1.83	n/a	26	42.6	9.65	243

Table 3: Coding workload, kept runs only (n = repetitions). "Identical" counts instances whose tokens equal the reference exactly, minimum over repetitions; zero-lane arms are compared with the plain model, and fixed-fold arms with the plain model padded to the same fold (which itself differs from the unpadded plain model on some instances, as any change of kernel does). Private-delta lanes are not expected to match. Footprint is the process high-water mark while serving.

Three lanes serve more than the plain model on every model (Figure 3, Table 3). The gain comes from decode: the plain model spends each decode step reading every weight once for one token, while a folded Hydra step reads each weight once for several rows, and continuous scheduling keeps those rows full. Prefill is not batched across lanes (each lane prefills alone), which bounds the gain on this prefill-heavy workload: 308,200 prompt tokens against about 14,000 generated. Section 6.1 takes the gap between capacity and speed-up apart.

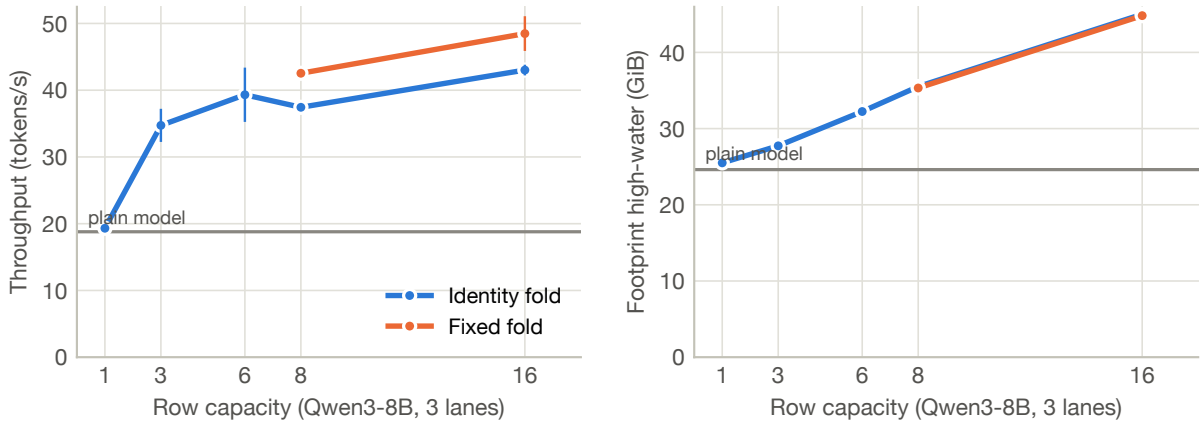

Figure 4: Qwen3-8B, three lanes, varying the number of rows a step may carry. Left: throughput (95% CI); right: process footprint high-water mark. The gray line is the plain model. Capacity 1 is Hydra used as a single stream.

Throughput rises with row capacity and costs memory for the extra concurrent caches (Figure 4). On Qwen3-8B, used as a single stream (capacity 1), Hydra runs at the plain model's speed: the per-row custody and authority checks cost under a millisecond per step. Past four rows the identity fold runs a second folded pass over the weights and the fixed fold runs one (Section 6.1), which is why the fixed fold leads at capacity 8 and 16.

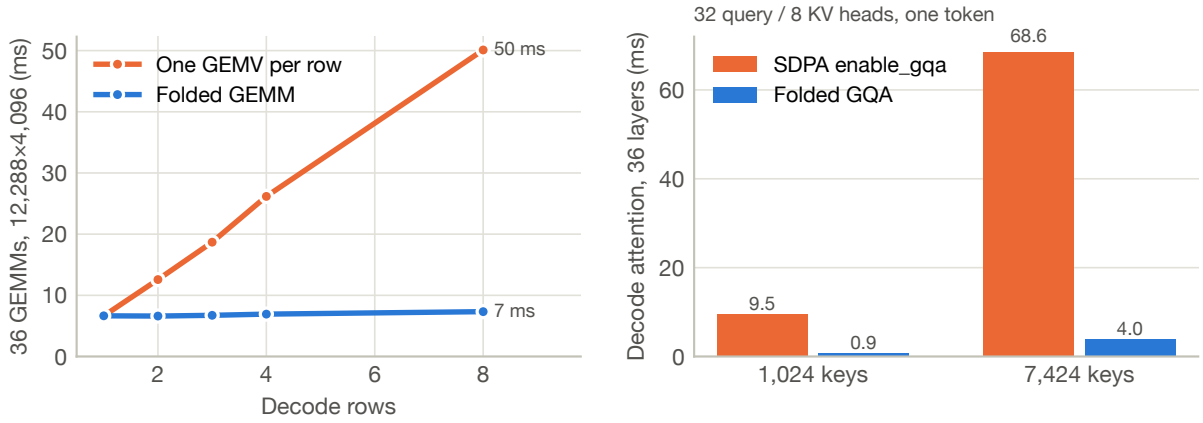

Figure 5: The two kernel pathologies that first made each lane cost a whole decode step (MPS, bf16). Left: 36 decode GEMMs of Qwen3-8B's feed-forward shape, one GEMV per row versus one folded GEMM. Right: decode attention over 36 layers, SDPA with `enable_gqa` versus folding each key-value group's query heads onto the query axis.

6.1 Why six rows are not six times faster

Decoding is sequential: a step emits one token per row, and the next token of a row cannot start until that step ends. The most a step can deliver is one token per row for the price of one, so capacity 6 would be 6× only if every step carried six rows, a six-row step cost no more than a one-row step, one Hydra row cost what the plain model's decode costs, and nothing but decoding took time. Each run logs the wall time and decoding row count of every scheduler step, which measures all four. With T_d the plain model's decode time per token (its wall time less its prefill), T_1 the median one-row decode step, $\bar{T}$ the mean decode-only step, S the decoding steps and D each engine's share of wall time spent decoding,

$$\frac{\text{Hydra tok/s}}{\text{plain tok/s}} = \underbrace{\frac{\text{tokens}}{S}}_{\text{tokens per step}} \times \underbrace{\frac{T_1}{\bar{T}}}_{\text{step cost}} \times \underbrace{\frac{T_d}{T_1}}_{\text{one-row path}} \times \underbrace{\frac{D_{\text{Hydra}}}{D_{\text{plain}}}}_{\text{prefill}} \tag{1}$$

exactly, not as a model: the four measured factors multiply to the measured ratio in every run (Table 4, Figure 6). For Gemma 3 4B at capacity 6 they are $5.30 \times 0.53 \times 0.73 \times 0.86 = 1.76\times$; for Gemma 3 12B, $5.19 \times 0.44 \times 0.95 \times 0.85 = 1.85\times$.

Tokens per step. 5.30 (4B) and 5.19 (12B) of 6 rows are filled on average: the 72-request burst ramps up at the start and its last requests drain alone at the end.

Step cost, the largest term. A step reads every shared weight once however many rows it carries, which is the whole gain, but it is not free per row. Each row's attention runs over its own cache (the isolation rule), and the identity fold is qualified at 4 rows on MPS bf16, so the fifth and sixth rows start a second folded pass over the weights. On Gemma 3 4B a decode step takes 39 ms with one row, 55 ms with four, 71 ms with five and 78 ms with six; on 12B, 75, 128, 164 and 187 ms. The kink at the fold width is visible in Figure 6 (left).

One-row path. A lone Hydra row costs more than the plain model's decode on the 4B model (39 ms against 28.2 ms) and about the same on 12B (75 ms against 71.3 ms): the Hydra path's per-row work does not grow with the model, so a larger model hides it. We have not broken this term down further.

Prefill. Each lane prefills alone, so Hydra spends about the same prefill seconds as the plain model, in a shorter run. The plain model spends 16% (4B) and 13% (12B) of its wall time in prefill; Hydra's larger prefill share costs a factor of 0.86 (4B) and 0.85 (12B) at capacity 6.

Without prefill the decode speed-up alone is 2.05× (4B) and 2.18× (12B) at capacity 6, the better guide for generation-heavy workloads. The fixed fold removes the fold-width cliff: one GEMM as wide as the row capacity, so the per-step gain keeps rising to 16 rows (Figure 6, dotted), at the price of matching the plain model run on the same fold rather than the unpadded one. What remains of the step cost grows with each row and includes that row's attention over its own cache, the part the isolation rule forbids sharing.

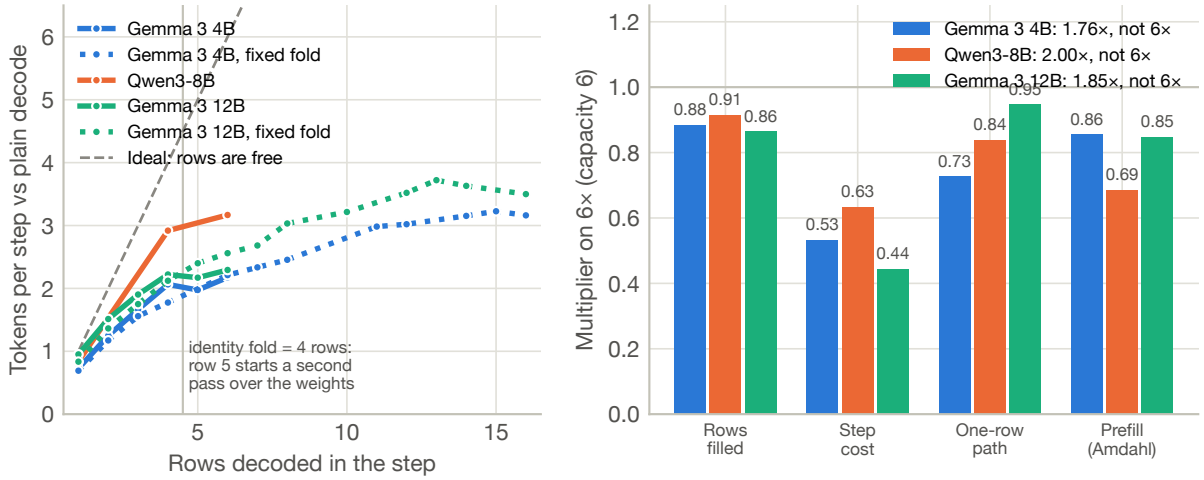

Figure 6: Why capacity 6 is not 6×. Left: tokens a decode step yields relative to the plain model's decode, against the rows the step carries (median step time per row count, from every step of the kept runs); the dashed line is rows at no extra cost. Past the identity fold width a row starts a second pass over the weights. Right: the four factors of Equation 1 at capacity 6; their product with 6 is the measured ratio.

Model	Configuration	Tokens per step	Step cost	One-row path	Prefill (Amdahl)	= Measured ratio	Decode only	Step, 1 row	Step, full
Gemma 3 4B	capacity 3	2.92	0.79	0.73	0.90	1.52×	1.69×	38 ms	47 ms
Gemma 3 4B	capacity 6	5.30	0.53	0.73	0.86	1.76×	2.05×	39 ms	78 ms
Gemma 3 4B	capacity 16, fixed fold	11.93	0.36	0.69	0.75	2.24×	2.98×	41 ms	143 ms
Gemma 3 4B	capacity 6, private deltas	5.25	0.54	0.73	0.87	1.78×	2.05×	39 ms	78 ms
Qwen3-8B	capacity 6	5.49	0.63	0.84	0.69	2.00×	2.91×	50 ms	80 ms
Qwen3-8B	capacity 6, private deltas	5.55	0.65	0.86	0.70	2.15×	3.07×	49 ms	77 ms
Gemma 3 12B	capacity 3	2.85	0.68	0.94	0.88	1.61×	1.82×	75 ms	108 ms
Gemma 3 12B	capacity 6	5.19	0.44	0.95	0.85	1.85×	2.18×	75 ms	187 ms
Gemma 3 12B	capacity 16, fixed fold	11.20	0.35	0.83	0.75	2.43×	3.25×	85 ms	326 ms
Gemma 3 12B	capacity 6, private deltas	5.31	0.44	0.91	0.86	1.83×	2.13×	79 ms	189 ms

Table 4: The factors of Equation 1 for every configuration with step timelines, means over kept repetitions. "Decode only" is the ratio with the prefill factor removed. Step times are medians over decode-only steps.

7 Memory footprint

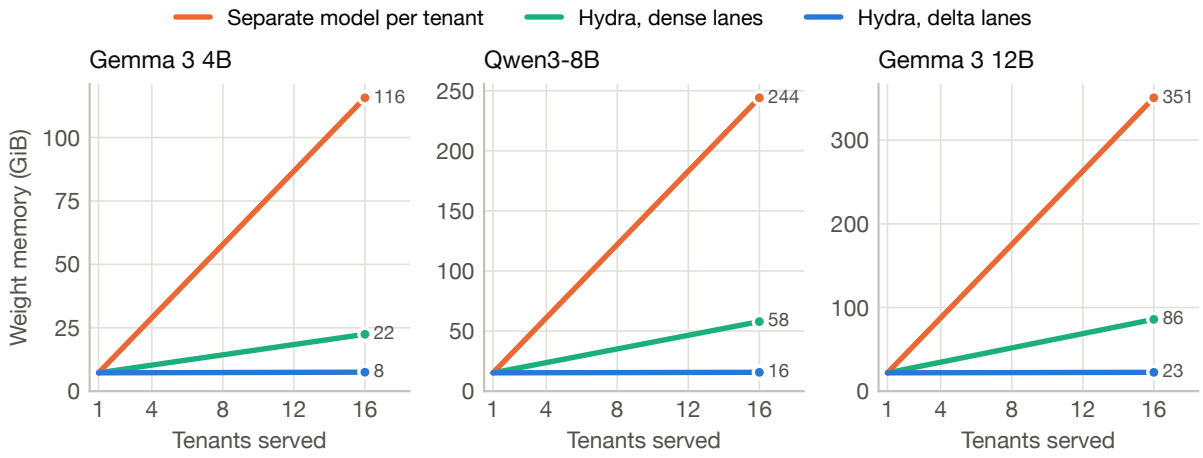

Figure 7: Weight memory to serve N isolated tenants: a model copy per tenant, Hydra with dense private attention per lane, and Hydra with rank-16 delta lanes. Computed from the parameter bytes each deployment reports.

Model	Model copy (GiB)	Dense lane (GiB)	Delta lane, rank 16 (MiB)	16 tenants, delta lanes (GiB)	16 model copies (GiB)
Gemma 3 4B	7.23	1.01	17.7	7.50	116
Qwen3-8B	15.26	2.84	29.8	15.72	244
Gemma 3 12B	21.92	4.26	41.9	22.57	351

Table 5: Parameter bytes per deployment, as reported by the running system. A delta lane holds its rank-16 factors on all four attention projections of every layer and its private norm gains.

A tenant's private state is its lane (Table 5): about 1% of a dense attention copy, and a fraction of a percent of a model copy. Sixteen tenants on delta lanes fit in the weight memory of one model (Figure 7).

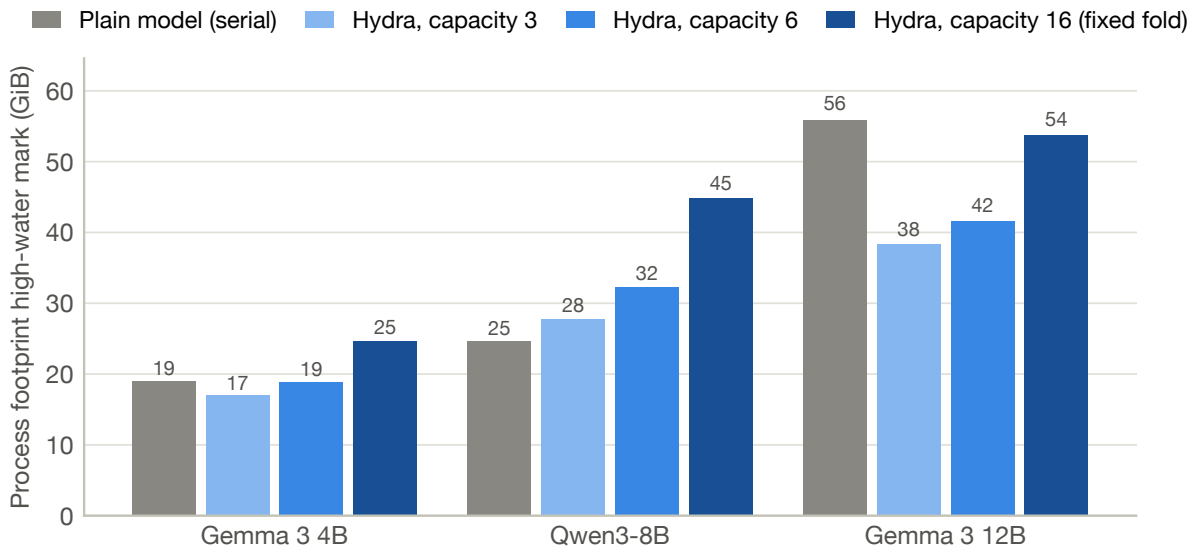

Figure 8: Process footprint high-water mark while serving the coding workload: everything the operating system charged the process, including model weights, key-value caches and allocator cache.

While serving, Hydra's footprint grows with the number of concurrent rows, since each row owns its key-value cache (Figure 8). A plain model serving three tenants in isolation needs three deployments, each with its own footprint; Hydra serves them from one.

8 Latency

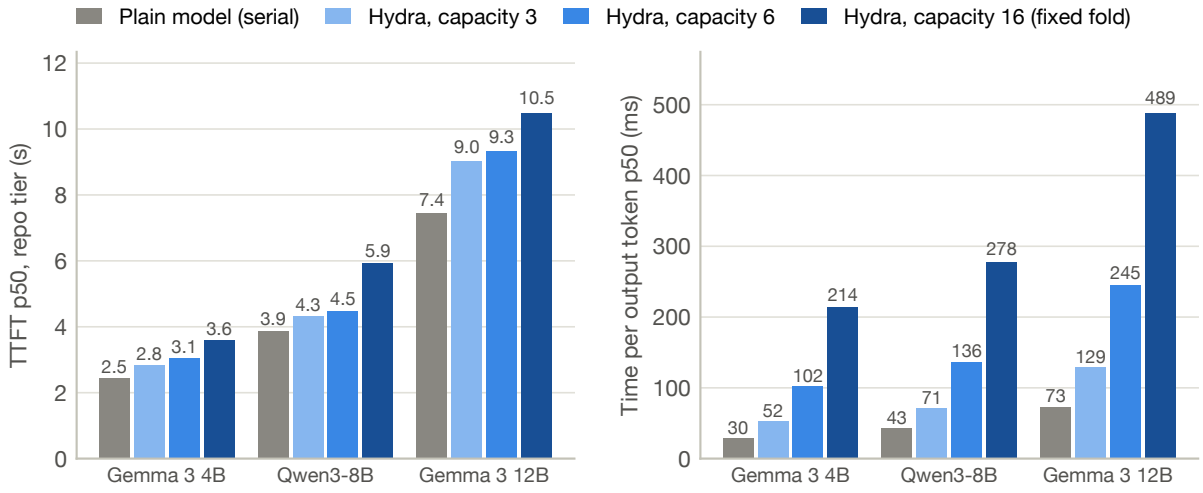

Figure 9: Left: time to first token for repository-tier prompts (about 7.5k tokens), median. Right: time per output token, median. Wider rows trade per-request decode speed for aggregate throughput.

Hydra trades per-request speed for aggregate throughput (Figure 9). Time to first token stays close to the plain model's because each lane prefills alone and the scheduler interleaves prefill chunks with decode; time per output token grows with the number of rows sharing a step. Because all 72 requests arrive at once, the aggregate gain also shortens how long the last request waits.

9 Authorization

Case (surface)	Gemma 3 4B	Gemma 3 12B
`foreign_intervention_owner_only` (instrumented driver)	pass	pass
`regime_0_success` (http service)	pass	pass
`regime_1_success` (http service)	pass	pass
`cross_request_hydra_batch` (http service)	pass	pass
`compound_success` (http service)	pass	pass
`two_requests_same_regime` (http service)	pass	pass
`foreign_regime_request` (http service)	pass	pass
`wrong_body_sha256` (http service)	pass	pass
`wrong_action` (http service)	pass	pass
`wrong_regime` (http service)	pass	pass
`wrong_model_id` (http service)	pass	pass
`wrong_runtime_id` (http service)	pass	pass
`wrong_principal` (http service)	pass	pass
`missing_authority` (http service)	pass	pass
`replay_rejected` (http service)	pass	pass
`expired_capability_rejected` (http service)	pass	pass
`operator_tenant_refused` (http service)	pass	pass
`invalid_workload_rejected` (http service)	pass	pass
`sidecar_loss_fails_closed` (sidecar process)	pass	pass
`tampered_private_artifact_refused` (cold start admission)	pass	pass
Total	20/20	20/20

Table 6: H2: a real server process over TLS with a real Gate sidecar, production enforcement, and signed capabilities. Wrong-fact cases each misbind one fact of a capability (body digest, action, lane, model, runtime, principal). The sidecar control kills the sidecar mid-service; the cold-start case flips bytes in a signed private artifact.

Table 6 runs the serving path end to end on Gemma 3 4B and 12B: authorized requests succeed with a sealed receipt per lane, two separately signed requests join one grouped batch with per-row custody, and every request that misbinds a fact, replays a capability, arrives expired, or names the operator tenant is refused before any private lane is touched.

10 Earlier evidence: other models, devices and designs

The results above are the current system on one machine. Earlier Hydra designs ran on other models, a speech architecture and CUDA hardware. Their measurements are summarized here, each read from its original study file; their throughput is not comparable with Section 6, but their exactness evidence is, and with this paper's it spans four model families (Qwen2.5, Qwen3, Gemma 3 and Whisper) from 0.5B to 12B parameters.

Model	Device	Design	What was compared	Result
Qwen2.5-0.5B-Instruct	NVIDIA L4	Prototype, dense lanes, R=4	1,024 prompt–lane pairs; 155.6M logits, 1.76B activations	max $\|\Delta\| = 0$
Qwen3-0.6B	CPU, fp32	Runtime, serial (H3)	4/4 lanes logits exact; 16 greedy steps	max $\|\Delta\| = 0$
Qwen3-4B	CPU, fp32	Runtime, serial (H3)	4/4 lanes logits exact; 8 greedy steps	max $\|\Delta\| = 0$
Qwen3-4B-Instruct	NVIDIA A100	Prototype, dense lanes, R=2	128 prompt–lane pairs; 19.4M logits, 1.18B activations	max $\|\Delta\| = 0$
Qwen3-4B-Instruct	NVIDIA A100	Prototype, dense lanes, R=4	256 prompt–lane pairs; 38.9M logits, 2.37B activations	max $\|\Delta\| = 0$
Qwen3-4B-Instruct	NVIDIA A100 80GB	Prototype lifecycle, R=8	448 full-vocabulary comparisons under foreign cancellation and replacement	448/448 exact
Qwen3-4B-Instruct	NVIDIA A100 80GB	Prototype lifecycle, R=16	896 full-vocabulary comparisons under foreign cancellation and replacement	896/896 exact
Qwen3-4B-Instruct	NVIDIA H200	Prototype lifecycle, R=32	1,792 full-vocabulary comparisons under foreign cancellation and replacement	1,792/1,792 exact
Qwen3-4B-Instruct, trained	NVIDIA H200	Prototype, R=8 donor interventions	256 donor mutations (256 effective)	1,792/1,792 receivers exact
Whisper tiny.en	CPU, fp32	Prototype, R=1 and R=4	500 utterances; 288 own-lane interventions	transcripts identical; 864/864 foreign exact

Table 7: Exactness across models, devices and designs. "Prototype" is the research implementation that preceded the Runtime (dense private attention per lane); "Runtime" is the serving system of this paper. Every comparison is exact equality of full-vocabulary logits or of transcripts.

Setting	Result
Qwen3-4B-Instruct, A100, prototype R=4 (eager)	prefill vs private attention run serially: 4.35× at 64 tokens, 2.19× at 256, 1.17× at 1,024; 256-step decoder 2.77×. Not byte-exact: 0/257 decoder steps exact, 15 argmax flips
Qwen3-4B-Instruct, prototype dense-lane ladder	R=8 33.9 GiB, 63.4 tokens/s (A100 80GB); R=16 62.0 GiB, 52.3 tokens/s (A100 80GB); R=32 118.3 GiB, 109.6 tokens/s (H200) (parameters plus packed snapshots)
Qwen3-8B, NVIDIA L4, Runtime 0.8.0, 4 dense lanes	22.85 vs 14.64 tokens/s for the plain model, 1.56× (before the kernels of this paper)
Whisper tiny.en, CPU, R=4	decoder 675 tokens/s vs 556 serial (1.21×); weights 236 MB vs 604 MB for four replicas; peak RSS 1.07 vs 1.41 GB

Table 8: Throughput and footprint from earlier designs, each against its own reference. The prototype's grouped execution was fast but not byte-exact; Whisper's decoder ratio is a ratio of medians.

What the earlier designs show. The research prototype established that private attention lanes could be exactly isolated when run one at a time (Table 7), on Qwen2.5, Qwen3 and Whisper, on L4, A100, H200 and CPU, up to 32 lanes and with trained lanes. Its grouped execution was fast, 4.35× serial private attention for short prefill, but not exact: batching changed the GEMM reduction order, so its decoder drifted and flipped argmaxes. Its lanes were dense, so memory grew by a full attention stack and a packed snapshot per lane. The Runtime closed both gaps: device-qualified folds made grouped execution exact (Section 3), and delta lanes cut a lane from gigabytes to tens of megabytes (Section 7). A separate study on the public Qwen3-4B-Instruct checkpoint (H200, bf16) reports 541 of 541 IFEval outputs identical to the native model under an isolated gate, a multiplexed gate and changed peers, in an audit of 340 shards whose raw archive is not retained on this workstation[2].

11 What the guarantee covers

The guarantee is exact within a stated surface: a lane's outputs and its authority. Outside it, by the same threat model as CDP [1], are a compromised host or process, process-memory extraction and hardware side channels. Timing is also outside it, and we measured it.

[2] `schemen-substrate/docs/research/HEAD_INDEPENDENCE_SCIENTIFIC_WRITEUP_2026-09-08.md`.

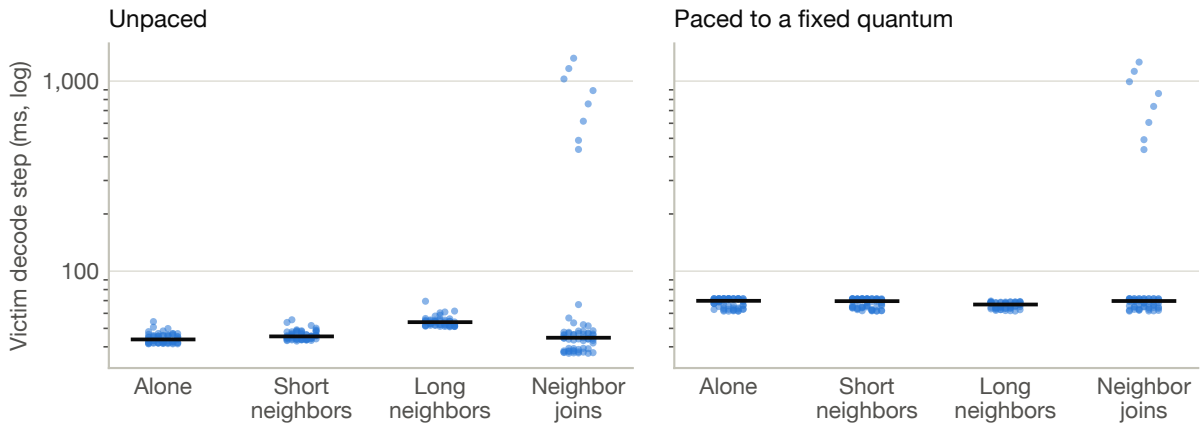

Figure 10: Timing channel on Qwen3-8B: a victim lane's decode step latency when alone, beside neighbors with short or long contexts, and when a long neighbor joins. Horizontal bars are medians. Pacing every step to a fixed quantum (right) costs about 55% of decode speed and does not remove the prefill stall.

A lane can tell from its own step latency whether its neighbors hold long contexts (AUC 0.99) and sees a neighbor's prefill as a stall of up to 1.3 s (Figure 10). What leaks is the shape of neighbors' work, never its content: outputs are byte-identical in every scenario. Tenants who must not learn each other's activity should be served on separate rosters; fixed-slot prefill and constant-work decode would close the channel at a throughput cost we have not measured.

12 Limitations

- **One device.** Every result is on one Apple M5 Max with MPS. Byte-exactness is a property of kernels on a device and must be qualified per device; CUDA qualification of the folded kernels, continuous scheduling and exactness remains open. An earlier CUDA run (NVIDIA L4, dense lanes, before this work's kernels) measured 1.56× the plain model with four lanes.
- **Constructed lanes.** Throughput uses zero lanes so that identity to the plain model is testable; private lanes are random perturbations, not trained tenants.
- **Repetitions.** Two to three repetitions per configuration; confidence intervals are wide for some arms.
- **Workload.** One synthetic coding workload with burst arrival. A 381-part journal-digest replay on Qwen3-8B (2.6 million prompt tokens) was token-identical to the plain model on 381 of 381 parts at 41.6 tokens/s, but has no clean plain-model timing baseline.
- **Prefill kernel.** The accelerator flash-prefill kernel supports head dimension 128; Gemma 3's 256 falls back to SDPA.
- **Receipts.** Custody receipts are content seals, not hardware attestations.

13 Conclusion

Isolation and efficiency are usually traded against each other in multi-tenant model serving. Hydra's lanes make isolation exact and make it cheap: one copy of the weights, a tenant's private state measured in tens of megabytes, and aggregate throughput above the plain model's on every model we tested, with every lane's output byte-identical to the plain model running that lane alone. The obvious next steps are CUDA qualification, trained tenant lanes, and a scheduler that closes the timing channel.

References

[1] Ryan R. Cryptographic Dimension Partitioning: exact gate-mask partitions for authenticated multi-tenant models. Schemen, 2026.

[2] E. J. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685, 2021.

[3] L. Chen et al. Punica: Multi-Tenant LoRA Serving. arXiv:2310.18547, 2023.

[4] Y. Sheng et al. S-LoRA: Serving Thousands of Concurrent LoRA Adapters. arXiv:2311.03285, 2023.

[5] G.-I. Yu et al. Orca: A Distributed Serving System for Transformer-Based Generative Models. OSDI, 2022.

[6] J. Juravsky et al. Hydragen: High-Throughput LLM Inference with Shared Prefixes. arXiv:2402.05099, 2024.

[7] H. He and Thinking Machines Lab. Defeating Nondeterminism in LLM Inference. Thinking Machines Lab: Connectionism, 2025.

[8] Gemma Team. Gemma 3 Technical Report. arXiv:2503.19786, 2025.

[9] Qwen Team. Qwen3 Technical Report. arXiv:2505.09388, 2025.

[10] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP, 2023. arXiv:2309.06180.

[11] L. Zheng et al. SGLang: Efficient Execution of Structured Language Model Programs. arXiv:2312.07104, 2023.

[12] L. Song et al. The Early Bird Catches the Leak: Unveiling Timing Side Channels in LLM Serving Systems. arXiv:2409.20002, 2024.

[13] Zheng et al. InputSnatch: Stealing Input in LLM Services via Timing Side-Channel Attacks. arXiv:2411.18191, 2024.

[14] Selective KV-Cache Sharing to Mitigate Timing Side-Channels in LLM Inference. arXiv:2508.08438, 2025.

A Reproducibility

Runtime commits: Qwen3-8B coding arms `bfd8990` (identity fold) and `3e9357a` (fixed fold); Gemma 3 arms `0cfc36d`; serial and HTTP acceptance `e9a052a`/`efdb2fa`. The evaluation harness, task generator and schedules are in `tools/hydra_acceptance/coding_eval/`; every figure and table in this paper is generated from `data/paper_data.json` by `collect.py`, `tables.py` and `figures.py` in `docs/research/hydra-paper/`. The per-paper research series in `docs/research/hydra-lane-efficiency/` documents each kernel and scheduler change with its own measurements.