

Holding Long Conversations in GPU Memory

Capacity engineering for a self-hosted, real-time guidance service on vLLM and Nemotron 3 Nano

Shawn Barnes · TensorSpace · October 2026

ABSTRACT

Advisory Guide is a customer support workflow tool that gives an agent short, grounded suggestions during a live conversation. Its serving cost is set by memory rather than compute: every session keeps a large, stable preparation prefix and a growing transcript resident on the GPU, while only a small fraction of sessions generate at any moment. We describe how that constraint was identified, the model selected for it, and how capacity was qualified on a single H100 PCIe 80GB with vLLM 0.17. Replacing a hybrid DeltaNet model with NVIDIA Nemotron 3 Nano 30B-A3B raised the reported KV pool from 374,752 to 2,973,696 tokens at matched FP8 cache precision (7.9x), and from 183,456 tokens against the earlier model's default 16-bit cache (16.2x). A residency sweep shows that prefix-cache reuse does not degrade gradually: hit rate held at about 83% up to the pool size and fell to exactly 0% between 1.06x and 1.20x of it. Admission was therefore expressed as a token budget at or below the pool, and a seven-hour soak at that budget served 69,312 requests across 200 resident sessions with zero errors and zero preemptions. We also report three operational findings that changed how the service is benchmarked and released: output length is a capacity parameter, a passing health check is not evidence of a working server, and a recoverable artifact is not a reproducible one.

Overview: The System in One Request

An agent loads preparation material into the desktop client before a conversation: product documentation, account notes, policies, or meeting notes. During the call, the client transcribes audio on the device. It does not interrupt with unrequested advice. The user presses Suggest when a moment matters, and only then does the system spend inference capacity. That product decision is also what shapes the workload: many sessions open at once, few of them generating in any given second.

On Suggest, the client assembles a single text request: the stable preparation first, then retained conversation history, then the newest transcript. A control plane validates the session and its entitlement and assigns a GPU worker. Each worker runs a gateway in front of vLLM that accepts only a short-lived signed grant scoped to that worker, so the client never holds a reusable credential. The answer streams back, and the client advances its transcript cursor only after a complete turn, so a failed generation never becomes lost conversation state.

Desktop: capture → local ASR → prompt assembly | Control plane: session, entitlement,
grant | Worker: gateway → vLLM → stream

Figure 1. Three cooperating systems. The same split is the first diagnostic step for any failed request: capture, admission, authorization, queueing, network, or generation.

1. Finding the Real Constraint

The binding limit is GPU memory held by sessions that are waiting. Measured preparation bundles in this product span 15K to 49K tokens, and each Suggest appends about 316 tokens of transcript and reply, almost perfectly linearly. A thirty-minute call at one press per minute roughly doubles a session from about 9,100 to about 18,600 tokens. Most of that state is idle at any instant. Two pressures therefore act independently: resident context occupies the KV pool whether or not the user is generating, and active requests consume compute and scheduler slots.

For attention layers the per-token cache cost is fixed by architecture:

$$\text{bytes per token} = 2 \times L_{\text{attn}} \times H_{\text{kv}} \times d_{\text{head}} \times \text{bytes per element}$$

where the factor of two covers keys and values. vLLM preallocates a pool of such entries at startup. When aggregate resident context exceeds the pool, cached prefixes are evicted, and the next request from an evicted session must recompute its entire prompt before producing a token. For a person mid-conversation that recomputation decides whether the suggestion is useful.

Two early investigations pointed at the request path before the hardware. First, the client waited far longer than the runtime reported spending on GPU work; comparing network, tokenization, request shape, and runtime topology located the delay outside the model. Second, latency grew as sessions accumulated, and the first proposal was more GPUs. Load testing showed the session memory was rewriting earlier context each turn, invalidating the cached prefix on every request. Making the prompt append-only, with stable preparation first and new turns only ever added at the end, restored reuse on the same hardware.

Finding. Measure where the time goes before buying capacity. In both cases the GPU was adequate and the request path was the problem.

2. Architecture Selected for That Constraint

We selected NVIDIA Nemotron 3 Nano 30B-A3B because it holds far less cache state per token than the candidate it replaced. Three properties contribute, and one commonly cited property does not.

Few attention layers. Most layers are Mamba-2 state-space blocks, which carry a fixed-size recurrent state per sequence instead of a cache that grows with context. Only a small number of layers use attention and need per-token keys and values. **Grouped-query attention.** Those layers share two key-value heads across their query heads, shrinking H_{kv} in the equation above. **FP8 cache.** Keys and values are stored at one byte per element. **Mixture of experts does not help memory.** The model routes each token through roughly 3B of about 30B parameters, which reduces compute per token, but every expert must stay resident, so none of the capacity result is attributed to it.

Capacity was read directly from the KV pool vLLM reports at startup on the same H100 PCIe 80GB, because that figure includes allocator overhead that a closed-form estimate omits. On vLLM 0.17 the hybrid's recurrent state is allocated per cache block alongside the attention pages, so the reported pool is the operative budget. Table 1 gives the comparison. At matched FP8 cache precision the ratio is 7.9x; the 16.2x figure includes the change from a 16-bit to an 8-bit cache.

Model and cache precision	Reported KV pool	Ratio vs. 183,456
Qwen3.6-27B (DeltaNet hybrid), FP8 weights, 16-bit KV	183,456 tokens	1.0x
Qwen3.6-27B (DeltaNet hybrid), FP8 weights, FP8 KV	374,752 tokens	2.0x
Nemotron 3 Nano 30B-A3B, FP8 weights, FP8 KV	2,973,696 tokens (42.57 GiB)	16.2x

Table 1. Reported KV pool on one H100 PCIe 80GB. Nemotron vs. Qwen at matched FP8 KV: 7.9x.

The base model was adapted with LoRA supervised fine-tuning for behavior rather than knowledge: answers that are brief, grounded in the user's loaded preparation, and usable while the user is still speaking. Facts arrive at run time in the prompt, which keeps them current without retraining.

Finding. Choose the model for the binding constraint. At matched cache precision the selected architecture holds 7.9 times more context per GPU.

3. Confirming It Under Load

3.1 Method

Item	Setting
Hardware	Lambda H100 PCIe 80GB, driver 570.148.08, CUDA 12.8
Software	vLLM 0.17.0, torch 2.10.0, transformers 4.57.6
Artifact	Fine-tuned Nemotron 3 Nano, FP8 weights and FP8 KV; 18 files verified by hash rollup
Cache	Prefix caching on; block size 4,224 tokens; pool 2,973,696 tokens
Workload	~15K-token sessions with append-only journals; rotated bursts of 10
Gates	Full response p50 ≤ 4.46 s, p90 ≤ 7.89 s, max ≤ 9.85 s, from a recorded real call

Table 2. Qualification environment. Results do not transfer to another SKU without remeasurement.

3.2 Residency: the cache is a step, not a slope

Resident context was raised from 0.25x to 2.0x of the pool while arrival was held near 1.08 requests per second (Figure 2). Below the pool, prefix hit rate held at about 83% within half a point across a fourfold range of session counts. That ceiling is structural: with 4,224-token blocks and ~14,900-token prompts, three full blocks are cacheable and the trailing partial block is always recomputed. The last clean point was 1.063x at about 81%; the first failing point, 1.197x, measured exactly 0.0%. This is the expected behavior of a least-recently-used cache under a rotating access pattern: once the working set exceeds capacity, each access evicts the entry needed next. The measurement locates where that edge falls for this workload and model. Latency then rose more gradually and flattened near 5.9 seconds. Overload degraded service without endangering the worker: at 2.01x with the cache gone, 300 of 300 requests succeeded with no preemptions, and only the median gate failed.

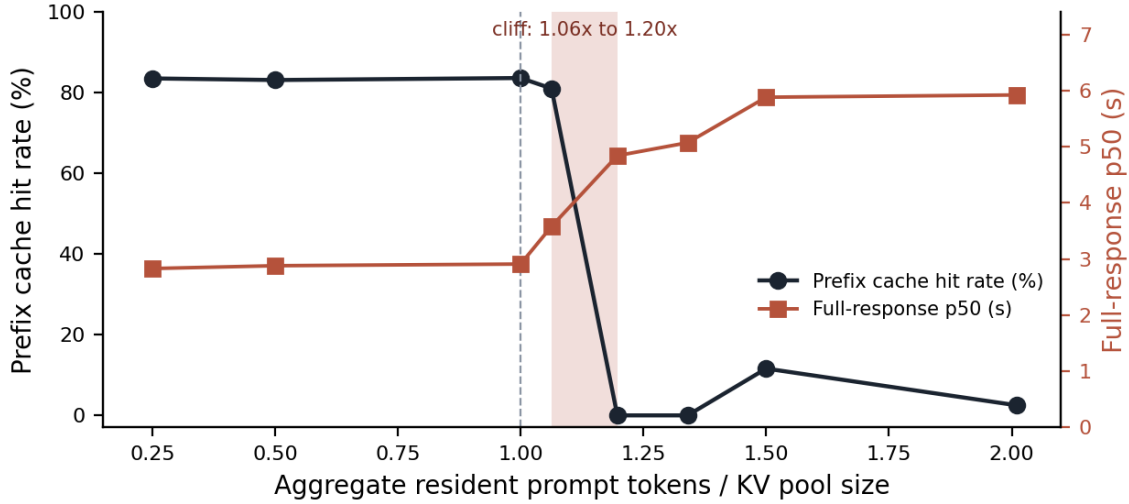


Figure 2. Prefix-cache hit rate and full-response median against resident tokens as a multiple of the KV pool. Points at 1.06x and 1.20x used a padded probe validated against real journals at 1.00x (TTFT p50 within 1%).

3.3 Rate: a second, independent limit

Holding residency near 1.05x and sweeping open-loop arrival found a separate ceiling at 3.0 requests per second (p50 4.36 s, p90 6.42 s). At 4.0 the queue grew to 42 and the median reached 11.5 seconds. Raising max_num_seqs from 16 to 64 did not move the gated ceiling, which shows compute saturation rather than a scheduler artifact, but it roughly halved overload latency (6.89 s versus 11.53 s at 4.0) and pushed divergence from 5 to 6 requests per second at no cost to the pool. For any realistic cadence of one press every 20 to 60 seconds, rate binds before residency.

3.4 Admission is a token budget

The admission rule that follows is to keep aggregate resident prompt tokens at or below the pool and arrival at or below 3.0 requests per second, whichever binds first. A session count is a derived number and is not portable: the same pool holds 198 sessions of 15K tokens but only 60 of 49K, and an even mix across the measured range lands near 91. Admission must also project growth. Two hundred sessions admitted at 9,087 tokens occupy 0.61x of the pool; with no new admissions they reach 1.25x after thirty minutes, so a worker admitted to budget at call start walks itself over the cliff on its own.

3.5 Sustained load

The qualified configuration then ran for 7.00 hours: up to 200 resident sessions at 2.75 requests per second, journals growing at the measured slope, continuous churn, and 5% of requests abandoned mid-stream. It served 69,312 requests with zero errors and zero preemptions; 2,624 sessions were created, 2,430 retired, 6 admissions were blocked, and all 3,440 mid-stream disconnects were absorbed. The waiting queue never exceeded 1, peak concurrent generation was 14, and the median ended 0.10 seconds faster than it began.

3.6 Output length is a capacity parameter

On the earlier Qwen configuration, two otherwise identical 30-minute runs (767 requests each, ten real preparation bundles of 7.5K to 43.6K tokens, Poisson arrivals with synchronized bursts) differed only in whether the production system prompt was included. Without the instructions that set answer length, median output was 169 tokens instead of 36, long generations held slots, peak in-flight reached 25 against a limit of 16, and the run failed every gate. With the real request, it passed.

Metric	Without production prompt	With production prompt
Full response p50 / p90 / worst	8.00 / 23.95 / 65.35 s	1.47 / 3.21 / 9.20 s
Output tokens p50	169	36
Peak in-flight requests	25	11
Prefix cache hit rate	87.7%	94.9%
Errors / preemptions	0 / 0	0 / 0

Table 3. One variable changed. The first run measured a missing prompt, not the GPU.

Finding. Capacity is a step. Admit by token budget below it, project session growth, and benchmark the real request, including the instructions that set output length.

4. Trust, Failure, and What Remains Open

Data path. Audio is transcribed on the device and never transmitted. Text does leave the device: preparation and transcript travel to the assigned worker, which is the boundary a privacy review should examine. The worker gateway verifies a short-lived grant before any generation, and the KV cache is transient computation state, discarded on eviction or restart.

Health is not capability. One server passed its health checks while every first generation failed, because a kernel compiled on first use required a build tool missing from the image. In another case a replacement process failed to bind its port while the previous process kept answering. Readiness now requires a representative inference and confirmation that the responding process reports the intended release; the environment is frozen by a hash rollup over all 18 artifact files.

Recoverable is not reproducible. The approved artifact was restored from object storage and matched its recorded hashes. A fresh export, however, came out byte-different because quantization calibration inputs had drifted. Retrieving a known release and rebuilding it from source are separate guarantees, and each now has its own test.

Limitations. All measurements are from one Lambda H100 PCIe and are non-certifying for other SKUs. The residency sweep used ~15K-token sessions, the bottom of the real distribution. The FP8 KV scales export as 1.0, which is the correct clamped value for the observed activation ranges, but answer quality with an FP8 cache has not yet been re-validated at full context length; the closing test is the same evaluation with the cache at FP8 and at 16-bit. The output-length experiment used a turn mix lighter than the reference call.

Finding. State what has been proven and what has not. Readiness must prove real work by the intended release, and every unverified setting stays listed until a test closes it.

Conclusion

Four steps recur through this work. Find the constraint that actually binds, which here was resident memory rather than model quality or raw compute. Act on it with a specific mechanism: an architecture that stores less per token and a prompt that never rewrites its own prefix. Confirm it with measurement under the real workload, which revealed that cache reuse fails as a step and that admission must be a growing token budget. Tell the limits plainly, so the next decision rests on evidence.

References

- [1] W. Kwon et al. Efficient Memory Management for Large Language Model Serving with PagedAttention. SOSP 2023.
- [2] T. Dao and A. Gu. Transformers are SSMS: Generalized Models and Efficient Algorithms Through Structured State Space Duality (Mamba-2). ICML 2024.
- [3] J. Ainslie et al. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. EMNLP 2023.
- [4] E. J. Hu et al. LoRA: Low-Rank Adaptation of Large Language Models. ICLR 2022.
- [5] NVIDIA. Nemotron 3 Nano 30B-A3B model card. Hugging Face, 2025.
- [6] vLLM project. Automatic prefix caching and FP8 KV cache documentation, v0.17.