



LLM Model Routing: Small vs Large Models Cost Model

Plan and build private AI compute with GPU Smith. We connect workload requirements with GPU hardware, networking, deployment and operating decisions so your infrastructure fits the work it must perform.

Published by GPU Smith · 2026-10-20 · llm model routing · llm routing cost model · small vs large language model cost · self-hosted llm cost per token · private llm fleet cost optimization · llm router quality threshold · cascade routing · on-premise llm inference cost · routellm · speculative decoding

Original article: <https://gpusmith.com/articles/en/small-vs-large-model-routing-cost-model>



In this article

1. Executive Summary
 2. Introduction and Background
 3. Key Changes in the Cost Model
 4. Evaluation Design and the Quality Constraint
 5. Capacity Design and GPU Residency
 6. Implementation Considerations and Process Changes
 7. Data Analysis and Evidence
 8. Implications and Future Directions
 9. Frequently Asked Questions (FAQs)
 10. Conclusion
-

Executive Summary

Routing requests between a small and a large language model is usually sold on per-token API prices, but a private fleet pays for whole GPUs, warm replicas and reprocessing, not for tokens. Published research reports large savings under its own experiments: the RouteLLM project reported cost reductions of over 85% on MT Bench, 45% on MMLU and 35% on GSM8K against a GPT-4-only baseline (www.lmsys.org), FrugalGPT reported savings from 50% to 98% depending on the dataset (arxiv.org), and the Hybrid LLM paper reported up to 40% fewer calls to the large model with no drop in response quality (arxiv.org). Those figures were priced per API token on public benchmarks, and the RouteLLM authors caution that real-world distributions may differ substantially (arxiv.org).

This report converts the question into a quality-constrained, whole-fleet cost model. Quality-constrained cost is minimized only among configurations that meet per-task acceptance thresholds and latency objectives, and the small-route saving must subtract router inference, escalations to the large model, retries and the fixed cost of keeping two pools warm. In a clearly labeled hypothetical example (placeholder throughputs, not measurements), an all-large pool needs 36 GPUs for a 100 requests-per-second peak, while a cascade that sends half of traffic to a small pool and escalates 10% of those requests needs 26 GPUs, a 27.8% reduction at an on-demand H100 list price of \$3.99 per GPU-hour ([lambda.ai](https://lambdai.com)). The same example shows break-even near a 20 requests-per-second peak, and shows that a router's apparent savings rise as it sends more hard queries to the small model, which is exactly the behavior a quality constraint must forbid.

A cascade or learned router can change output quality, whereas speculative decoding, under its stated algorithm, produces the same output distribution as the target model (arxiv.org). Google's speculative-cascades research frames the distinction: cascades often reach better cost-quality trade-offs, while speculative decoding offers a guarantee of quality-neutrality (arxiv.org).

The practical recommendation is to decide with an organization-owned labeled set, task-stratified results with uncertainty intervals, and deterministic rules and an all-large pool as baselines. NIST guidance says AI systems should be tested before deployment and regularly while in operation (nvlpubs.nist.gov). Residency

adds a second constraint: weights need roughly 2 GB of GPU memory per billion parameters at 16-bit precision (huggingface.co), so the large pool dictates GPU count long before the router does. Revalidate after any model, router, policy or traffic change.

Introduction and Background

Enterprise platform teams that serve a heterogeneous private workload face a recurring decision: keep one large model resident and size every request for it, or run a small model beside it and decide per request which one answers. Most public guidance prices the choice per token; this report instead treats it as an infrastructure engineering problem. It is written from the perspective of GPU Smith, an independent private AI infrastructure engineering consultancy that advises on GPU procurement, workload sizing, inference optimization and air-gapped deployments, and that neither sells chips nor operates a cloud.

Three terms need precise definitions. A **router** makes one decision before generation, choosing which model answers. A **cascade** runs the small model first and defers to the large model when a confidence or verification signal fails. **Speculative decoding** uses a small draft model to propose tokens that the large model verifies; under the exact algorithm it speeds inference while preserving the target distribution ([paper](#)). RouteLLM, an open-source framework, trains routers that choose between a stronger and a weaker model during inference (arxiv.org). The RouteLLM code is published under an Apache License (github.com), the blog reports that matrix factorization achieved 95% of GPT-4 performance using 26% GPT-4 calls (www.lmsys.org), and its README describes routers based on a weighted Elo calculation (github.com) and on a BERT classifier trained on preference data (github.com).

The research literature is strong on method and thin on private hardware. RouteLLM priced its pair at an assumed \$24.7 and \$0.24 per million tokens (arxiv.org), a roughly 100-fold gap that no self-hosted pair is guaranteed to reproduce. RouterBench's negative comparison with Zero concerns its KNN/MLP predictive-router experiments in Section 5.3. The same paper reports simulated cascades outperforming Zero with sufficiently accurate judges and improvements over Zero for all tested routers in its separate RAG experiment ([paper](#), Sections 5.3–5.4). Google's speculative-cascades research helps define the cost and quality distinction but is not a deployment benchmark for any particular fleet (research.google).

The decision boundary for this report is therefore: given a task mix, a per-task quality floor, a privacy requirement and latency service-level objectives (SLOs), does operating two model pools cost less than operating one, once residency, queueing, escalation and quality risk are counted?

Key Changes in the Cost Model

From per-token price to whole-GPU footprint

API comparisons multiply tokens by a price per million tokens. A private fleet pays for GPUs that exist whether or not they are busy. The memory rule of thumb from Hugging Face's documentation is that loading a model with X billion parameters requires roughly $2 \times X$ GB of GPU memory in bf16 or fp16 (huggingface.co), so an 8-billion-parameter model needs about 16 GB at 16-bit precision and about 8 GB at 8-bit, while a 70-billion-parameter model needs about 140 GB or 70 GB. Quantization trades model precision for a smaller memory

footprint (docs.vllm.ai), a trade that is itself a quality risk. A mixture-of-experts model such as Qwen3-235B-A22B has 235B parameters in total and 22B activated (huggingface.co); compute per token follows the activated count, but weight residency follows the total, which is why a sparse large model can be cheap to run per token and still expensive to keep warm.

The PagedAttention paper found that existing systems used only 20.4% to 38.2% of KV cache memory to store token states (arxiv.org), so KV cache capacity, not weights alone, bounds concurrency, and vLLM pre-allocates a share of GPU memory for it (docs.vllm.ai).

From average accuracy to quality-constrained cost

A single aggregate score can hide a regulated or high-risk task class. The cost model therefore minimizes cost only among configurations that satisfy per-task acceptance thresholds and SLOs, and it reports task-stratified results with uncertainty intervals. The quality estimate is the critical input: the unified routing and cascading paper found that a strategy will fail if the quality estimates are inaccurate (arxiv.org).

From "cascade equals speculative decoding" to a clear distinction

The two are routinely conflated. Speculative decoding was introduced as a way to make exact decoding from large models faster, and the guarantee of an identical output distribution holds for any choice of approximation model. A draft model must be served in addition to the target, and in the DeepMind experiments it was trained with the same tokenizer as the target (arxiv.org), so tokenizer compatibility is a deployment constraint. Its benefits depend on method and load: vLLM documents both latency and high-QPS throughput gains, with results depending on the model family, traffic, hardware and sampling settings (documentation). Benchmark the chosen method at the intended load.

Cascades and routers differ in kind. Google's research describes cascades as offering better cost-quality trade-offs, often even outperforming the large model, while speculative decoding offers a guarantee of quality-neutrality. Google's blog on the hybrid approach states that cascades allow for some variability in quality (research.google). A cascade can therefore make an answer better, worse or merely different, and the organization must measure which; NIST likewise says AI systems should be tested before deployment and regularly while in operation (nvlpubs.nist.gov).

From a static pool to a pair of pools with different failure modes

Two pools mean two sets of replicas, two upgrade paths, two sets of cold-start behavior and a router in front. Router and verifier overhead must also be measured: it should be negligible compared with model computation time, which makes it an acceptance test for the router.

Table 1 below compares the approaches a platform team can choose between. It is a decision aid, and the right-hand column states what must be measured on the organization's own traffic rather than what any paper reported.

Approach	Can it change answer quality?	Extra capacity and fixed cost	Evidence in the literature	What to measure locally
All-large baseline	No relative to the chosen large model	One pool sized to peak for every request	Reference point for every published saving	Qualified throughput per replica at the latency SLO
Deterministic rules (task type, tenant, length, keyword)	Yes, for misrouted tasks	Two pools, no learned-router inference	RouterBench's KNN/MLP predictive-router experiment does not establish performance for these task, tenant, length or keyword rules	Per-task acceptance rate on the labeled set
Learned router (for example RouteLLM matrix factorization)	Yes, and it depends on the training distribution	Two pools plus router inference and embedding -service capacity	Matrix factorization reached 95% of GPT-4 performance with 26% GPT-4 calls (www.lmsys.org) on public benchmarks	False-small rate by task class, threshold sweep, router and embedding egress. RouteLLM's mf and sw_ranking require an OpenAI API key for embeddings even with local answering models (README); adapt and validate the embedding path for fully private or air-gapped operation, and include any local embedding service in capacity and cost
Cascade with verifier	Yes, and a weak verifier lets errors through	Two pools, reprocessing of escalated requests, verifier cost	FrugalGPT returns a generation if the score is higher than a threshold (arxiv.org)	Escalation rate, verifier miss rate, added latency
Speculative decoding	Not under its stated algorithm	Draft model memory, same tokenizer	Identical output distribution under the exact algorithm; vLLM documents latency and throughput gains that depend on method and workload (docs.vllm.ai)	Acceptance rate and speedup at your load

Speculative decoding preserves the target distribution under the exact algorithm and primarily targets inference speed. Treat cost savings as conditional: measure whether lower GPU time or fewer required replicas outweigh drafting and verification costs ([vLLM documentation](#), [drafting and verification overhead](#)). The interpretive consequence is that the two pools in a cascade are a governance commitment as well as a capacity commitment. The remaining sections turn each column into a procedure.

Evaluation Design and the Quality Constraint

Build a representative, organization-owned labeled set

RouteLLM trained on roughly 80K Chatbot Arena battles (arxiv.org) and augmented them with a judge-labelled preference dataset of approximately 120K samples costing around \$700 (arxiv.org). That data reflects public chat preferences, not a regulated enterprise's tickets, contracts or code. The authors acknowledge that real-world applications may have distributions that differ substantially from these benchmarks; the reported results are research results, not guarantees for a private workload. The labeled set should be sampled from production traffic with privacy controls, stratified by task class, and refreshed on a schedule.

A workable design uses these elements:

- **Task taxonomy:** each request carries a class label (for example summarization, extraction, code, policy question, regulated advice) and a risk tier.
- **Acceptance rubric:** per-class criteria that a reviewer can apply, such as factual match, schema validity, citation presence and prohibited-content checks.
- **Gold labels:** human review of a stratified sample, with a second reviewer on high-risk classes.
- **Paired outputs:** for every item, store the small model's output and the large model's output so the router and cascade can be replayed offline.
- **Frozen provenance:** record weight hashes, quantization format, tokenizer, serving engine version, prompt template and sampling parameters for both pools.

If an automated judge supplements human review, its limitations must be tracked. The MT-Bench study reported that strong judges can match human preferences well, achieving over 80% agreement (arxiv.org), and also documented position, verbosity and self-enhancement biases (arxiv.org). A judge is a measurement instrument with its own error rate and should be calibrated against the human labels, not treated as ground truth.

Cost-sensitive confusion matrix and uncertainty intervals

Define the router's decision against the ground-truth label of whether the small model's answer meets the acceptance rubric. The confusion matrix then has four cells with unequal costs:

- **Small-correct (true small):** the request is routed small and passes. The saving is realized.
- **False-small:** the request is routed small and fails the rubric. The cost is a quality failure if undetected, or an escalation plus reprocessing if a verifier catches it.
- **False-large:** the request is routed large though small would have passed. The cost is lost saving only.
- **True-large:** the request is routed large because it needed to be.

The false-small rate, not overall accuracy, is the quantity a quality constraint governs, and it must be reported per task class. Report each rate with a confidence interval. The NIST Engineering Statistics Handbook documents the Wilson score interval for a proportion and notes that its quality does not strongly depend on the sample size or the proportion (www.itl.nist.gov) and that its lower limit cannot be negative (www.itl.nist.gov),

which matters for rare failure rates near zero. Bootstrap resampling is a complement; the handbook says 500 subsamples is usually sufficient (www.itl.nist.gov). A class with few labeled examples has a wide interval, which is a reason to route it to the large pool until more labels exist.

The threshold sweep

RouteLLM exposes a cost threshold per request, and its README says a higher cost threshold leads to lower cost but may lead to lower-quality responses (github.com). It also states that the threshold controls the cost-quality tradeoff and is calibrated on a dataset (github.com). The research measures quality as the share of the weak-to-strong performance gap that a router recovers. A private sweep should replace "performance gap recovered" with the per-class acceptance rate and plot, for each threshold, the escalation share, the undetected-failure share per class, time to first token (TTFT) and inter-token latency (ITL), and pool utilization. TTFT is the time from sending a request to receiving its first streamed output (docs.vllm.ai), and NVIDIA defines ITL as the average time between consecutive tokens (docs.nvidia.com). The selected threshold is the cheapest point that satisfies every per-class constraint, not the point with the best aggregate score.

Capacity Design and GPU Residency

Required replicas are ceilings of measured throughput

For each pool, the required replica count is the ceiling of peak demand routed to that pool divided by the measured qualified throughput per replica, plus spare replicas for failure and maintenance. A model that is too large for a single GPU but fits on a single node is served with tensor parallelism (docs.vllm.ai), which is why the example below uses two-GPU replicas for the large model. Qualified throughput means the request rate a replica sustains while meeting the latency SLO and the acceptance rubric; it must be measured on the organization's prompts and output lengths with the exact serving configuration. Little's law connects the quantities: MIT lecture notes state that if any one of the four expected queue values is known the others can be determined (ocw.mit.edu), so mean in-flight requests equal arrival rate times mean time in system. The probability that an arriving request must wait in a pool of identical servers is given by the Erlang C formula (archive.nptel.ac.in), though continuous-batching engines are not ideal servers, so it is only a first approximation.

Queues also amplify tail latency. The Tail at Scale shows that if each server has a one-second 99th-percentile latency and a request fans out to 100 servers, 63% of user requests will take more than one second (www.barroso.org). Google's site reliability guidance advises small queue lengths relative to the thread pool size (sre.google). For a cascade, escalation is a second hop, so the large pool's tail latency adds to the small pool's and the end-to-end SLO must be tested on escalated requests specifically.

Co-residency, partitioning and cold loads

Four residency patterns are available, with different failure and utilization properties:

- **Separate whole-GPU pools:** simplest to reason about and to roll back, at the cost of stranded capacity when the small pool is lightly loaded.

- **Hardware partitioning with Multi-Instance GPU (MIG):** a GPU can be securely partitioned into up to seven separate GPU instances (docs.nvidia.com), with L2 cache banks, memory controllers and DRAM address busses assigned uniquely to each (docs.nvidia.com). The H100 SXM supports up to seven MIGs at 10GB each (www.nvidia.com), and the H200 up to seven at 18GB each (www.nvidia.com); compute instances inside a GPU instance share memory and engines (docs.nvidia.com). A 10 GB slice cannot hold an 8-billion-parameter model at 16-bit precision (about 16 GB by the rule above), so partitioning suits small models, not mid-sized ones, unless larger slices are used.
- **Software sharing with Multi-Process Service (MPS):** MPS is useful when each application process does not generate enough work (docs.nvidia.com), and it can limit the memory allocatable by CUDA APIs (docs.nvidia.com), but a fatal fault is reported to all the clients running on the subset of GPUs (docs.nvidia.com).
- **Engine-level sharing with memory fractions:** vLLM's `gpu_memory_utilization` is the fraction of GPU memory to be used for the model executor (docs.vllm.ai), will use the default value of 0.92 if unspecified (docs.vllm.ai), and pre-allocates GPU cache using that percentage of memory (docs.vllm.ai). Two engines on one GPU must split the fraction.

The following arithmetic gives pre-overhead upper bounds, not measured KV capacities. At an illustrative `gpu_memory_utilization` of 0.92, an 80 GB GPU budgets about 73.6 GB to one engine. Subtracting roughly 35 GB of 8-bit weights per GPU for a two-GPU 70B-class replica leaves at most 38.6 GB before other memory costs. If an 8B-class 8-bit model is co-hosted on each GPU with fractions of 0.80 for the large engine and 0.12 for the small engine, the corresponding weight-only remainders are 29 GB and 1.6 GB per GPU. These estimates exclude peak activations, runtime and non-PyTorch allocations, CUDA graphs and any quantization metadata or workspace overhead; vLLM profiles memory to determine cache capacity, and CUDA graphs consume additional memory ([worker documentation](#), [memory guidance](#)). Compare the observed KV allocation, supported concurrency and latency from each exact serving configuration; these subtractions do not establish a 25% reduction in available KV cache. When KV space runs short, vLLM can preempt requests to free up KV cache space for other requests (docs.vllm.ai). Measure whether co-residency's GPU savings outweigh preemption and tail-latency costs.

Cold loads are the other penalty. vLLM's sleep-mode write-up describes the case of two LLMs that each fit on a GPU but not both at once (vllm.ai), and says reloading models on demand costs 30-100+ seconds per switch (vllm.ai), while sleep mode is 18-200x faster than a full reload (vllm.ai). Sleep level 1 offloads weights to CPU memory and discards the KV cache (docs.vllm.ai); level 2 discards both the weights and the KV cache (docs.vllm.ai). Concurrent loading can significantly improve loading times from object storage or network file shares (docs.vllm.ai). For a 70 GB 8-bit checkpoint, the weight-transfer floor is 70 s at 1 GB/s, 14 s at 5 GB/s and 2.8 s at 25 GB/s before engine initialization, and a pool that must absorb a 50 requests-per-second stream during a 60-second load queues about 3,000 requests. That is why the cost model treats scale-to-zero for the large pool as a quality-of-service decision, not a free saving.

Implementation Considerations and Process Changes

Operating procedure

A reproducible procedure follows from the sections above:

- **Freeze the baselines:** record the all-large and all-small acceptance rates per class, plus deterministic-rule routing, on the labeled set.
- **Replay the router offline:** for each threshold, compute the confusion matrix per class with Wilson intervals, then discard thresholds that breach any class constraint.
- **Measure qualified throughput:** run each pool at its candidate configuration and record the rate at which the TTFT and ITL SLOs hold. Automatic prefix caching reuses the KV cache of existing queries (docs.vllm.ai), so measure with production prompt prefixes.
- **Size with ceilings:** compute replicas per pool, add spares, and add the router's own footprint.
- **Publish the scorecard:** one row per task class with labeled count, small-route acceptance rate and false-small rate (each with a Wilson interval), escalation rate, p95 TTFT and ITL, GPUs, cost per million tokens and a pass or fail against the class threshold.
- **Run a shadow period:** route a copy of live traffic through the candidate and compare against the large model's answers on a sample reviewed by humans.
- **Stage the rollout:** begin with low-risk classes, keep regulated classes on the large pool, and expand as intervals allow.

Drift, fallback, audit logs and rollback

Routing quality is distribution-dependent, so drift monitoring is part of the cost. Google's Rules of Machine Learning suggest measuring the difference between performance on "next-day" data and the live data (developers.google.com). In practice that means re-scoring a rolling labeled sample each week, tracking the escalation rate and the router score distribution, and alerting when either moves outside a pre-agreed band.

Fallback behavior must be explicit. If the router or the small pool is unhealthy, traffic should fail over to the large pool, which means the large pool needs headroom or the system needs load shedding. A router that silently concentrates traffic on the large pool during a small-pool outage creates an overload condition, so the large pool's failover capacity must be sized and tested rather than assumed.

NIST's AI Risk Management Framework calls for tracking emergent risks based on actual performance in deployed contexts (nvlpubs.nist.gov) and for post-deployment monitoring plans (nvlpubs.nist.gov). The Generative AI Profile points to change management records, version history and metadata as useful for incident response (nvlpubs.nist.gov) and to procedures for escalating incidents to the organizational risk management authority (nvlpubs.nist.gov). For a router, that translates into a per-request audit record containing the request class, router score, chosen pool, model and router versions, escalation outcome and verifier verdict, plus the ability to roll back the router, the threshold or the small-model version independently. A vLLM-based semantic router paper lists audit logging among the controls for a strict privacy scenario (arxiv.org), which fits air-gapped deployments where logs must stay on premises.

Licensing and provenance

Open-weight pairs carry license terms that belong in the decision record. Meta's Llama 3.1 model card describes a custom commercial license, the Llama 3.1 Community License (huggingface.co), with a separate-license requirement under Section 2 if, on the Llama 3.1 version release date, the products or services made available by or for the licensee or its affiliates had more than 700 million monthly active users in the preceding calendar month ([Section 2 of the license](#)), while the Llama 3.3 70B card describes a corresponding community license agreement (huggingface.co) and the Qwen3-8B repository carries the Apache License, Version 2.0 (huggingface.co). Record weight hashes and quantization formats for both pools so an evaluation can be replayed exactly.

Data Analysis and Evidence

Published results versus what they measure

The principal published routing and cascade results are summarized below with the conditions under which each was measured. They are research results under specific experiments and should not be read as expected savings for a private fleet.

- **RouteLLM blog:** cost reductions of over 85% on MT Bench, 45% on MMLU and 35% on GSM8K at 95% of GPT-4 performance (www.lmsys.org), using GPT-4 Turbo as the strong model and Mixtral 8x7B as the weak model (www.lmsys.org).
- **RouteLLM paper:** savings were computed at an assumed price ratio of roughly 100 to 1 per million tokens, so a different price ratio changes every saving, and the reported router overhead is a cost ratio rather than a GPU-seconds figure.
- **FrugalGPT:** savings from 50% to 98% depending on the dataset, with learning the cascade itself consuming resources.
- **Hybrid LLM:** up to 40% fewer calls to the large model with no drop in response quality, expressed in call counts rather than GPU hours.
- **Unified routing and cascading:** combining the two strategies, with gains that depend on accurate quality estimators.

API-level savings require separate accounting from whole-fleet savings. Section 3 of the unified routing and cascading paper explicitly sums the individual model costs in a cascade ([paper](#)). Retain those cumulative stage charges, and model private-fleet idle capacity, weight residency and cold loads separately.

A worked, hypothetical whole-fleet model

(Hypothetical Example) The numbers below are placeholders chosen to demonstrate the arithmetic; they are not measurements of any model, GPU or vendor, and every one must be replaced with a locally measured value.

- **Demand:** 100 requests per second at peak, 35 requests per second on average, 1,100 tokens per request (800 input, 300 output).

- **Large pool:** 70B-class model at 8-bit precision, two GPUs per replica, qualified throughput of 6 requests per second per replica.
- **Small pool:** 8B-class model, one GPU per replica, qualified throughput of 36 requests per second per replica, already net of the router's co-located inference.
- **Router:** one additional GPU when any traffic is routed small.
- **Escalation:** 10% of small-routed requests are detected as failing and are reprocessed on the large pool.
- **Spares:** one extra replica per pool.
- **Price:** \$3.99 per GPU-hour, the on-demand rate Lambda lists for an NVIDIA H100 SXM 80 GB in an 8-GPU instance ([lambda.ai](https://aws.amazon.com/lambda/pricing/)), over 730 hours per month. This is a rental proxy for a cost of ownership, which a buyer should replace with amortized hardware, facility and operations cost.

Large-pool load is the large-routed share plus escalations, $(1 - s) + 0.1s$ of peak, where s is the small-routed share. Small-pool load is s . *Table 2* below applies the ceiling rule.

Small-routed share (s)	Large replicas (2 GPUs each, incl. spare)	Small replicas (incl. spare)	Total GPUs (incl. router)	Monthly cost at \$3.99/GPU-hour	Saving vs all-large
0 (all-large)	18	0	36	\$104,857	0%
0.3	14	2	31	\$90,294	13.9%
0.5	11	3	26	\$75,730	27.8%
0.7	8	3	20	\$58,254	44.4%
0.9	5	4	15	\$43,690	58.3%

The grid shows savings that scale with small-routed share, which is the pattern the research reports, but it omits the constraint that matters. Each row assumes the small model's answers pass the acceptance rubric. A higher s is only admissible if the labeled set shows the extra routed requests are ones the small model handles. At the 100 requests-per-second peak, the 0.5 row saves \$29,127 per month relative to the baseline. With the same assumptions, the cascade becomes cheaper than all-large near a 20 requests-per-second peak (at 55% small share, 10 GPUs versus 9), is slightly more expensive at a 10 requests-per-second peak (6 versus 7 GPUs, because of the router and a minimum two-replica small pool), and is 33% cheaper at 100 requests per second. Below the break-even, a single pool is the better choice.

At average demand, the same fleet costs roughly \$1.04 per million tokens in the all-large configuration and \$0.75 per million in the 0.5 configuration (101 billion tokens per month at 35 requests per second, 1,100 tokens each). Both figures reflect a peak-sized pool at about 35% average utilization. For comparison, a fully utilized large replica in this example would cost about \$0.34 per million tokens and a fully utilized small replica about \$0.03, so utilization, not model size alone, drives a self-hosted cost per token. The API list prices in public price tables show large ratios that a self-hosted pair need not reproduce: OpenAI's pricing page lists gpt-6-luna at \$0.10 input and \$0.50 output per million tokens ([developers.openai.com](https://openai.com/pricing/)), gpt-6.1-sol at \$2.00 input and \$10.00 output ([developers.openai.com](https://openai.com/pricing/)), and gpt-6-astra at \$10.00 input and \$50.00 output

(developers.openai.com), a 100-fold ratio between the last and the first, and Anthropic's pricing documentation warns that newer tokenizers produce approximately 30% more tokens for the same text (platform.claude.com), so per-token prices from different families are not directly comparable.

Energy is a second-order input in this example. An H100 SXM has a maximum thermal design power of up to 700W (www.nvidia.com); the H100 NVL is rated 350-400W (www.nvidia.com). A 700 W GPU at the Uptime Institute survey's global average power usage effectiveness (PUE) of 1.54, quoted on Google's data-center page (datacenters.google), draws about 1.08 kW at the facility level, which at the US industrial average of 8.91 cents per kilowatt-hour for the rolling twelve months ending July 2026 (www.eia.gov) costs about \$70 per GPU-month. That is about 2.4% of the \$2,913 monthly rental rate, though an owner will see a different split. Google reports a 2025 fleet-wide average PUE of 1.09 for its own data centers (datacenters.google), and a PUE of 2.0 means that for every watt of IT power, an additional watt is consumed (datacenters.google), so facility design moves this term strongly.

Sensitivity to easy-query share and false-small rate

Table 3 below varies the share of truly easy queries (e) and the router's false-small rate (phi, the share of hard queries sent to the small pool). It assumes the router sends 90% of easy queries small, the small model fails 3% of those and the failure is detected, and a verifier detects half of the hard queries wrongly sent small. All other inputs are as above.

Easy share (e)	False-small rate (phi)	Small-routed share	Total GPUs	Monthly cost	Undetected failures, share of all traffic
0.3	0.05	30.5%	31	\$90,294	1.75%
0.3	0.15	37.5%	30	\$87,381	5.25%
0.5	0.05	47.5%	26	\$75,730	1.25%
0.5	0.15	52.5%	24	\$69,905	3.75%
0.7	0.05	64.5%	20	\$58,254	0.75%
0.7	0.15	67.5%	20	\$58,254	2.25%

The table makes two points. First, a higher easy-query share is the dominant driver of saving, moving the fleet from 31 to 20 GPUs. Second, a higher false-small rate lowers cost or leaves it unchanged, by sending more traffic to the cheap pool, while it multiplies the undetected failures. The two easy-share 0.7 rows both cost \$58,254, yet if the quality constraint allows at most 1% undetected failures only the phi 0.05 row qualifies, and at easy share 0.5 the cheaper row exceeds the limit by nearly four times. That is the sense in which the cost frontier is quality-constrained: unconstrained cost minimization selects the riskiest router.

Three further sensitivities should be run in the spreadsheet. **Load correlation:** if the easy share falls from 0.5 to 0.3 at peak, large-pool demand rises from about 55 to about 72 requests per second against 66 requests per second of capacity (ten active replicas plus one spare) sized at the design mix, so a mix shift at peak is a capacity breach even with unchanged total volume. **Escalation storms:** raising escalation from 10% to 20%

and 40% of small-routed traffic moves the fleet from 24 to 26 and 30 GPUs at a 55% small share.

Partitioning: the memory arithmetic above shows that co-hosting trades GPUs for KV capacity, which the replica ceilings must reflect.

Implications and Future Directions

For an enterprise AI platform lead, the model supports a decision rule instead of a recommendation. Operate two pools when the measured easy-query share is high enough, the peak is above the break-even for the organization's own qualified throughputs, the false-small rate meets every per-class constraint with an upper interval bound inside the tolerance, and the team can run, monitor and roll back both pools. Keep one pool when the volume is near or below break-even, when the high-risk classes dominate, or when the labeled set cannot yet support per-class intervals. A deterministic rules baseline deserves a real trial because rules are easier to audit.

Several developments could shift the balance. Sleep-mode style switching narrows the cold-load penalty for models that do not fit together (docs.vllm.ai), and open routing tools continue to appear, such as the NVIDIA LLM Router blueprint for intent-based routing that matches user intents to specific models (github.com). Speculative decoding can be combined with either pool to cut latency without altering answers under its stated algorithm, but it adds draft-model memory and does not reduce the number of pools.

Governance is the lasting cost. A router creates a new decision point whose errors are invisible without sampling, so the audit record and the periodic re-scoring are recurring operational work. GPU Smith's advisory view, as an independent consultancy rather than a vendor of the compared options, is that the evaluation set and the residency arithmetic should be built before any procurement commitment, and that the spreadsheet should be re-run after every model, router, policy or traffic change.

Frequently Asked Questions (FAQs)

What is an LLM routing cost model?

It is a calculation of total fleet cost for a routing policy: GPUs per pool at measured qualified throughput, router inference, escalations, spares and warm capacity, all subject to per-task acceptance thresholds and latency SLOs.

How do I route requests between small and large models?

Classify each request before generation using deterministic rules or a learned router, send it to the pool its class allows, and optionally verify the small model's answer and escalate failures. RouteLLM routes by a cost threshold (github.com), and FrugalGPT's cascade returns a generation if the score is higher than a threshold.

What is an LLM router quality threshold?

It is the router-score cutoff separating small-routed from large-routed requests. Sweep it on the labeled set and pick the cheapest point that satisfies every per-class constraint, with Wilson intervals on each false-small rate (www.itl.nist.gov).

What is cascade routing of small and large LLMs?

It combines an upfront routing decision with post-hoc checks after a model has run, and its value depends on the accuracy of the quality estimator.

How does small versus large language model cost compare, and what is the self-hosted cost per token?

API tables can show large ratios, such as the 100-fold gap between two listed gpt-6 models (developers.openai.com). On a private fleet, cost per million tokens equals pool GPU-hour cost divided by tokens served at the SLO. In the hypothetical example, a fully utilized large replica cost about \$0.34 per million tokens, about 12 times a small replica, and the all-large fleet cost about \$1.04 at average demand, so utilization matters as much as model size.

How should an on-premise inference cost model treat power and cooling?

Multiply GPU thermal design power (up to 700W for an H100 SXM (www.nvidia.com)) by facility PUE and the local tariff, then compare with amortized hardware cost. In the example, energy was about 2.4% of the rental rate.

How does private LLM fleet cost optimization differ from API optimization?

Private fleets pay for idle capacity, so optimization targets utilization, right-sized replicas, cold-start avoidance and partitioning while preserving per-class quality. API optimization targets tokens and price tiers.

Conclusion

Small versus large model routing can lower the cost of a private fleet, but the saving is conditional on facts that only the organization can measure: the share of easy queries, the router's false-small rate by task class, the qualified throughput of each pool and the cost of keeping both pools warm. Published results show that cascades and learned routers can reach large API cost reductions under their experiments, and the authors state that real distributions may differ. A whole-fleet model replaces those percentages with ceilings on measured throughput, and a quality constraint prevents the cheapest configuration from being the worst one.

The distinction between routing and speculative decoding should stay explicit in every design document: routers and cascades may change answers and require evaluation and governance, whereas speculative decoding preserves the target distribution under its stated algorithm and mainly changes latency. A sound rollout uses an organization-owned labeled set, task-stratified results with uncertainty intervals, deterministic and all-large baselines, a shadow period, audit records and a tested rollback path. Revalidate after any change to the models, the router, the policy or the traffic.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. <https://www.lmsys.org/blog/2024-07-01-routellm>
2. <https://arxiv.org/pdf/2305.05176>

3. <https://arxiv.org/abs/2404.14618>
4. <https://arxiv.org/html/2406.18665>
5. <https://lambda.ai/pricing>
6. <https://arxiv.org/pdf/2211.17192>
7. <https://arxiv.org/abs/2405.19261>
8. <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf>
9. https://huggingface.co/docs/transformers/main/en/llm_tutorial_optimization
10. <https://arxiv.org/abs/2406.18665>
11. <https://github.com/lm-sys/RouteLLM/blob/main/LICENSE>
12. <https://github.com/lm-sys/RouteLLM>
13. <https://arxiv.org/pdf/2403.12031>
14. <https://research.google/blog/speculative-cascades-a-hybrid-approach-for-smarter-faster-llm-inference>
15. <https://docs.vllm.ai/en/latest/features/quantization/index.html>
16. <https://huggingface.co/Qwen/Qwen3-235B-A22B>
17. <https://arxiv.org/pdf/2309.06180>
18. <https://docs.vllm.ai/en/latest/configuration/optimization.html>
19. <https://arxiv.org/pdf/2410.10347>
20. <https://arxiv.org/pdf/2302.01318>
21. https://docs.vllm.ai/en/latest/features/speculative_decoding
22. <https://vllm.ai/blog/2026-08-23-speculative-decoding-amd-gpus>
23. <https://arxiv.org/abs/2306.05685>
24. <https://www.itl.nist.gov/div898/handbook/prc/section2/prc241.htm>
25. <https://www.itl.nist.gov/div898/handbook/eda/section3/bootplot.htm>
26. <https://docs.vllm.ai/en/latest/benchmarking/cli>
27. <https://docs.nvidia.com/nim/benchmarking/llm/latest/metrics.html>
28. https://docs.vllm.ai/en/latest/serving/parallelism_scaling.html
29. <https://ocw.iti.hr/courses/civil-and-environmental-engineering/1-225j-transportation-flow-systems-fall-2002/lecture-notes/lec8queues.pdf>
30. <https://archive.nptel.ac.in/content/storage2/courses/117103017/module3/lec2/1.html>
31. <https://www.barroso.org/publications/TheTailAtScale.pdf>
32. <https://sre.google/sre-book/addressing-cascading-failures>
33. <https://docs.nvidia.com/datacenter/tesla/mig-user-guide/introduction.html>
34. <https://www.nvidia.com/en-us/data-center/h100>
35. <https://www.nvidia.com/en-us/data-center/h200>
36. <https://docs.nvidia.com/datacenter/tesla/mig-user-guide/concepts.html>
37. <https://docs.nvidia.com/deploy/mps/when-to-use-mps.html>
38. <https://docs.nvidia.com/deploy/mps/appendix-tools-and-interface-reference.html>

39. https://docs.vllm.ai/en/latest/configuration/engine_args.html
40. <https://docs.vllm.ai/en/v0.10.2/api/vllm/worker/worker.html>
41. https://docs.vllm.ai/en/latest/configuration/conserving_memory
42. <https://vllm.ai/blog/sleep-mode>
43. https://docs.vllm.ai/en/latest/features/sleep_mode
44. https://docs.vllm.ai/en/latest/models/extensions/runai_model_streamer.html
45. https://docs.vllm.ai/en/latest/features/automatic_prefix_caching.html
46. <https://developers.google.com/machine-learning/guides/rules-of-ml>
47. <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf>
48. <https://arxiv.org/html/2603.04444v1>
49. <https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct>
50. https://github.com/meta-llama/llama-models/blob/main/models/llama3_1/LICENSE
51. <https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct>
52. <https://huggingface.co/Qwen/Qwen3-8B/blob/main/LICENSE>
53. <https://developers.openai.com/api/docs/pricing>
54. <https://platform.claude.com/docs/en/about-claude/pricing>
55. <https://datacenters.google/efficiency>
56. https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=table_5_03
57. <https://github.com/NVIDIA-AI-Blueprints/llm-router>

THE PUBLISHER

About GPU Smith

GPU Smith provides independent engineering and research for private AI infrastructure. We help technical buyers and operators reason about GPU workload sizing, hardware selection, procurement, deployment and inference operations, from the compute system to the networking, power and cooling requirements around it.

Start with the workload

Useful infrastructure decisions begin with the models, data, throughput, latency and operating constraints a team actually has. GPU Smith helps connect those requirements with system design and deployment choices. The goal is a privately controlled AI environment whose capacity and operational demands are understood before a purchasing decision.

Hardware and supplier research

Our public reference library covers GPUs, complete systems and networking components. The server-vendor directory and manufacturing research help teams investigate suppliers, compare options and follow sources behind technical claims. We distinguish manufacturer specifications from measured benchmarks and advertised capabilities from tested configurations. Reference pages are research resources, not a live inventory listing or a binding equipment quote.

Deployment and operations

GPU Smith's engineering scope includes infrastructure integration, networking, power, cooling and the practical operation of inference workloads. Our published articles explain the tradeoffs behind deployment, procurement and ongoing operations so teams can ask better questions and document decisions.

Work with GPU Smith

Explore the [hardware library](#), [GPU references](#), [networking references](#) and [vendor research](#). [Contact GPU Smith](#) with your workloads and deployment constraints to discuss a project and confirm current scope, pricing and availability.

Inclusion of a manufacturer or product in our research does not imply a partnership, certification or endorsement.

Website: <https://gpusmith.com>

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.