Think Before You Grid-Search: Floor-First Triage for LLM Serving

Yihua Liu
liuyihua1994@gmail.com

July 2026

Abstract

Production teams optimizing large language model (LLM) serving face a configuration space spanning parallelism layout, batching, quantization, sparse attention, and kernel-level work. The common operational response is to benchmark many configurations and open heavy profilers whenever a latency target is missed. This paper argues for a different workflow: build an analytical floor first, reconcile benchmarks against that floor, and escalate to profiling only when the residual justifies it. *Estimation is the analytical layer of profiling*: without it, optimization degenerates to grid search.

We present Floor First, a residual-driven triage methodology with a zero-dependency artifact: a floor calculator plus an agent skill that makes the discipline enforceable in agentic optimization loops. The account is compositional—new attention or state-space variants enter by declaring one module, not by rewriting the framework—so the workflow applies to any autoregressive transformer served on accelerators. Floor First models each decode step as a five-dimensional resource vector (HBM bytes, FLOPs, network bytes, network messages, KV capacity). Terms that use the same resource add; independent resources can overlap. This gives two numbers: an optimistic floor, max, and a no-overlap floor, sum. Where a measurement falls inside this $[\max, \mathrm{sum}]$ interval is already a diagnostic: it tells us how much overlap the system is getting before any profiler is opened. Deployment alternatives are then compared by *wall ordering*—which resource wall binds first as load grows—rather than by point benchmarks.

As a case study, we analyze a DeepSeek-V3.2-style 671B MoE/MLA model on 16 NVIDIA H20 GPUs, a hardware point whose ridge point of ∼74 FLOP/byte (versus ∼590 for H100) makes it an extreme decode-oriented part that no published analysis characterizes. The floors show that TP16 decoding at batch 64 and 8K context is KV-capacity-limited to ∼70 concurrent requests, that DSA-style sparse attention removes the KV-bandwidth term but not the capacity wall, and that an EP16+DP-attention layout trades slightly worse same-batch weight traffic for an order-of-magnitude higher capacity wall (∼644 requests)—while, at cluster-calibrated communication constants, single-stream latency favors TP by 2.4×. The judgment between layouts is thus a computable function of the operating point, which explains why production deployments on identical hardware have shipped opposite attention layouts.

1 Introduction

In 2025, production teams serving DeepSeek-family models made opposite parallelism choices on comparable hardware. The official DeepSeek deployment decodes with large-scale expert parallelism and *data-parallel* attention [11]; vLLM's default DeepSeek recipe now enables the same layout [36]; yet a joint LMSYS–Ant Group deployment on the same 16×H20-96G configuration we study decodes attention with TP16 [24]. None of these reports derives the choice from an explicit account of hardware resources; the engineering community converged by experiment and folklore. This paper's position is that the disagreement was predictable from a small resource account—roughly five numbers per GPU plus the model dimensions—and that making this account systematic gives a general triage workflow for serving optimization.

The operational problem is broader than one layout choice. Serving optimization spans model variants, hardware SKUs, tensor parallelism (TP), expert parallelism (EP), data-parallel (DP) attention, prefill/decode disaggregation, continuous batching, sparse attention, quantization, CUDA graphs, communication libraries, and kernels. Trying combinations on production hardware is expensive, and a benchmark number alone is ambiguous: a slow TPOT can mean a fundamental bandwidth limit, an implementation gap, exposed communication, host overhead, or an underperforming kernel. Existing research offers increasingly accurate analytical models [4, 9, 27], simulators and configuration search [2, 39, 40], and serving

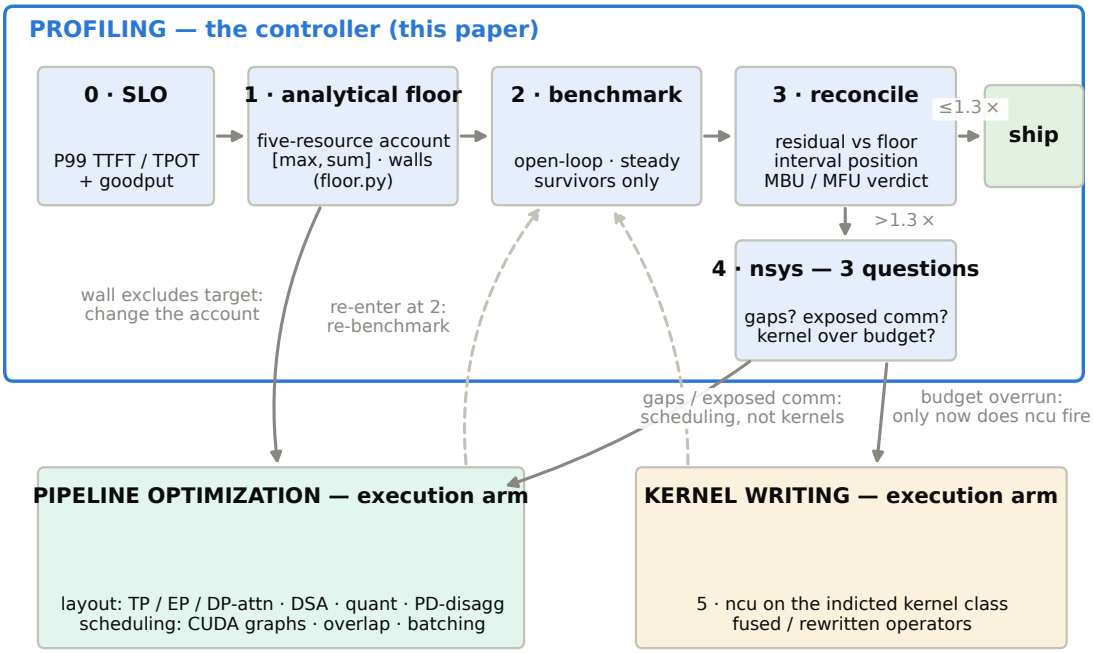

Figure 1: The serving-optimization loop. The floor model decides whether the next action should be layout/scheduling work, kernel work, or no profiling at all. Capacity and wall-ordering verdicts can rule out layouts before GPU time is spent. Nsight Systems is used only when the benchmark leaves a residual that the floor cannot explain; Nsight Compute is reserved for kernel classes that exceed their budget. Every change is re-measured and fed back into the loop.

systems [1, 26, 46]. What day-to-day performance engineering lacks is a *decision procedure*: when is a measured number close enough to the floor that profiling is wasted effort, and when it is not, where should the profiler look first?

Floor First fills that gap. The workflow computes per-step floors from model dimensions, calibrated hardware rates, parallelism, and workload shape. It then reads a benchmark as a *residual*: the gap between the measured time and the floor. If the residual is small, the investigation stops. If it is large, profiling has a specific job: check whether time is missing between kernels, whether communication is exposed, or whether a kernel class exceeded its analytical budget. The gap is sharpest for LLM coding agents, whose default optimization behavior is precisely the loop this paper's title warns against; the workflow therefore ships as an enforceable agent skill as well as a human checklist.

This draft makes five contributions:

1. **A residual-driven triage workflow** that separates target definition, floor construction, benchmark reconciliation, and profiler escalation, with explicit stopping rules—and whose verdicts dispatch the two execution arms of optimization, pipeline work and kernel work (§2.3, Figure 1).

2. **A two-sided floor model.** Summing contention within a hardware resource and maximizing across resources gives an optimistic floor; the plain sum gives a pessimistic one. The measurement's position inside $[\max, \text{sum}]$ is a zero-cost overlap diagnostic that existing analytical models discard by fixing an overlap assumption (§3).

3. **Wall ordering as an output.** Deployment alternatives are compared by which wall—KV capacity, KV bandwidth, weight bandwidth, compute—binds first as load grows, rather than by same-batch latency alone. Capacity is a first-class resource, not a post-hoc memory check (§5.3).

4. **Metric discipline.** A roofline identity (MFU/MBU = $I_{\text{work}}/I_{\text{ridge}}$) dictates which utilization metric is meaningful per phase; we systematize practitioner thresholds for triage verdicts and state their calibration protocol (§2.2, §2.4).

5. **A worked case study and artifact.** We give the first ridge-point characterization of the export-grade H20 GPU and a complete decode account for a DeepSeek-V3.2-style model on 16 H20s, deriving the TP-versus-EP+DP judgment, single-stream physical bounds, and the effect of sparse attention—as executable, zero-dependency Python plus an agent-readable skill (§5).

Status of this draft. Case-study numbers are analytical floors computed by the artifact; communication constants are cluster-calibrated (§4), GPU rates are datasheet unless marked otherwise. Section 8 states the measurement campaign required before submission; the residual sweep in particular is planned, not yet performed. A subsequent revision will fold that sweep back into §5: measured-versus-floor reconciliation plots with per-point $[\max, \text{sum}]$ intervals, the interval-position distribution, and an empirically grounded (or revised) escalation threshold.

2 Background and Motivation

2.1 Serving Metrics and Benchmark Discipline

LLM serving has at least three operational metrics: time to first token (TTFT), time per output token (TPOT), and goodput—throughput that satisfies a service-level objective (SLO), usually stated on tail latency. Floor First requires an explicit target before any optimization:

maximize goodput subject to P99 TTFT and P99 TPOT SLOs.

Without it, benchmark numbers are not even partially ordered.

Benchmarks that feed the workflow must obey four rules. (i) *Open-loop load generation*: closed-loop generators issue the next request only after the previous completes, so an overloaded system automatically receives less load and its tail latency looks healthy. This bias was formalized for classical systems by Schroeder et al. [30]; LLM serving evaluations rarely revisit the distinction, yet capacity claims are meaningless without it. (ii) *Realistic length distributions*, since the prefill:decode ratio and chunked-prefill interference depend on them. (iii) *Steady state*: before the KV pool fills, preemption never fires and prefix caches over-hit; the first minutes of a run are honeymoon numbers. (iv) *Tail percentiles*, because SLOs live at P99, not the mean.

2.2 Prefill and Decode Have Different Bottlenecks

Prefill processes many prompt tokens in parallel at high arithmetic intensity and is usually compute-bound; decode streams weights and KV cache per generated token and is usually bandwidth-bound at realistic batch sizes. This asymmetry underlies phase-splitting systems [1, 26, 46] and is the basis for metric choice.

Let workload arithmetic intensity be I_{work} = FLOPs/bytes and the hardware ridge point $I_{\text{ridge}} = F_{\text{peak}}/B_{\text{HBM}}$ [37]. With model FLOPs utilization MFU = FLOPs/$(t\,F_{\text{peak}})$ and model bandwidth utilization MBU = bytes/$(t\,B_{\text{HBM}})$ [8],

$$\frac{\text{MFU}}{\text{MBU}} = \frac{I_{\text{work}}}{I_{\text{ridge}}}$$

holds identically. The two utilizations are not independent; their ratio is fixed by the workload. For the case study of §5, decode at batch 64 has $I_{\text{work}} \approx 11$ FLOP/byte against an H20 ridge of 74: when MBU reaches 80%, MFU *equals* 12%. Low decode MFU is workload geometry, not a tensor-core bug. The metric-selection principle is: *report the utilization whose ceiling is 100% for the binding resource*—MBU for bandwidth-bound decode, MFU for compute-bound prefill.

2.3 The Workflow: One Controller, Two Execution Arms

Production inference optimization has three arms: *profiling* (knowing where time goes and how far from the limit it is), *kernel writing* (driving a single operator to the hardware limit), and *pipeline optimization* (parallelism layout, batching, overlap, scheduling). In this paper, profiling is not just "open Nsight." It is the decision loop that decides whether kernel work or pipeline work is worth doing. Kernel work without that decision can optimize an operator that is not on the critical path; pipeline tuning without it becomes grid search. Figure 1 shows the loop. A capacity or wall-ordering verdict can send us directly to a new layout before we spend GPU time. A residual caused by poor overlap or scheduling sends us to the pipeline arm. Only a kernel class that exceeds its budget sends us to Nsight Compute and kernel work. After each change, we benchmark again and compare the new result with the floor.

Within the controller, Floor First treats instrumented profiling as exception handling, not as a pipeline stage:

1. Define the objective: P99 TTFT, P99 TPOT, goodput.
2. Compute analytical floors and capacity walls for candidate configurations; discard configurations whose walls exclude the target operating point.
3. Benchmark the survivors under the discipline of §2.1.
4. If the measurement is within the escalation threshold of the floor (1.3× in our practice), stop—or move to a different optimization axis.
5. Otherwise open Nsight Systems and answer exactly three questions: are there gaps between kernels (host/scheduling/launch)? is communication exposed rather than overlapped? which kernel class exceeds its analytical budget?
6. Open Nsight Compute only for the kernel class that the budget table indicts, after gap and overlap causes are excluded.

2.4 Triage Thresholds and Their Status

Our operating rules: decode MBU $> 70\%$ means the implementation is near the floor on the current axis; 40–70% suggests overlap or scheduling recovery, worth an Nsight Systems pass; $< 40\%$ suggests a system-level defect (host-bound execution, missing CUDA-graph coverage, interference) that per-kernel work will not fix. For MoE prefill we lower the MFU bands to 50%/25% because all-to-all communication and expert imbalance are structural.

These numbers systematize practitioner experience [8] rather than derive from first principles, and we state their epistemic status plainly: they are *calibration targets*, not constants of nature. The calibration protocol is to collect (predicted floor, measured, post-hoc root cause) triples across configurations and check two properties: below the threshold, Nsight sessions should yield no actionable finding (no false negatives); above it, a root cause should be identifiable (no wasted escalations). §8 lists this as the first required experiment; until then the thresholds should be read as defaults that worked in our deployments, in the same spirit that Top-Down analysis shipped with fixed decision-tree cutoffs.

3 Analytical Model

3.1 Resource Vector and the Two-Sided Floor

Each decode step consumes a five-dimensional resource vector

$$r = (\text{HBM bytes},\ \text{FLOPs},\ \text{network bytes},\ \text{network messages},\ \text{KV capacity}).$$

The first four convert to time via calibrated rates; the fifth bounds feasible batch size and enters the goodput judgment (§3.5). Two aggregation rules follow from the hardware, not from convention: terms contending for the *same* resource add (weight reads and KV reads share HBM; bandwidth cost and per-message latency

share the NIC), while *independent* engines (HBM, SMs, NIC) run concurrently under perfect overlap, so only the slowest shows:

$$t^{\text{opt}} = \max(t_{\text{HBM}},\ t_{\text{compute}},\ t_{\text{network}}), \qquad t^{\text{sum}} = t_{\text{HBM}} + t_{\text{compute}} + t_{\text{network}}.$$

We keep both numbers because the gap between them is useful. If a measured step lands near t^{opt}, the hardware engines are already overlapping well and the next gain must come from reducing the dominant account itself. If it lands near t^{sum}, the system is behaving as if the engines were serialized, so scheduling and overlap are the first suspects. If it lands above t^{sum}, overlap cannot explain the measurement; time is leaking into costs outside the model, such as host gaps, preemption, or stragglers. Existing analytical models collapse this information by choosing an overlap assumption up front (e.g., compute hidden under memory with communication fully exposed [9]); the interval keeps overlap as something the benchmark can reveal. One caveat is structural: at batch 1 the layer-serial dependency chain leaves little to overlap, so the honest single-stream floor is t^{sum}, not t^{opt} (§5.6).

3.2 Decode FLOPs

Per token, parameter GEMMs cost $\approx 2P_{\text{act}}$ FLOPs, where P_{act} is the activated parameter count ($P_{\text{act}} = P$ for dense models); attention score/value products add a term that depends on context length and attention architecture:

$$\text{FLOPs}_{\text{decode}} \approx 2P_{\text{act}}B + \text{FLOPs}_{\text{attn}}(B, S).$$

In the decode regimes we study, attention FLOPs matter for the compute row, but the attention path is still dominated by reading KV from HBM. The FLOP account is therefore used mainly to check whether compute can ever become the wall (§5.3).

3.3 Decode Bytes: the MoE Union Correction

Decode HBM traffic per step is weight traffic plus KV-read traffic. For MoE models the two accounts diverge: FLOPs follow *activated* parameters, but weight bytes follow the *union* of experts activated across the batch, because a weight tile is read once per step regardless of how many tokens use it. Under uniform routing with E routed experts and top-k selection, the expected distinct-expert fraction per layer is

$$f_{\text{expert}}(B) = 1 - \left(1 - \tfrac{k}{E}\right)^{B}, \qquad f_{\text{expert}}(B) \le \min(1,\ kB/E),$$

so a batch of 64 touches $\approx 87\%$ of experts for E=256, k=8, and the conservative min bound saturates. This closed form was previously employed to analyze speculative-decoding gains for MoE [18] and in-batch expert sharing [35]; we apply it to deployment-sizing byte accounts, where it corrects both the weight-bandwidth wall and the compute-wall location by the MoE sparsity ratio (§5.3). Load imbalance makes the uniform-routing expectation optimistic; §7 discusses the failure mode.

KV bytes depend on attention architecture and sharding. For GQA, the KV cache partitions across up to n_{kv} heads under TP. For MLA-style latent attention the latent KV is shared by *all* heads: head-parallel TP cannot shard it, so every TP rank stores and reads the full cache—the observation first stated in engineering channels [31, 36] and in recent architecture work [25, 34, 43], whose responses are to redesign the attention or its sharding. We take the deployment-side view: given unmodified MLA, the replication cost makes DP attention the structurally favored high-concurrency layout, and §5 quantifies by how much.

3.4 Communication

TP decode performs two all-reduces per layer; with ring all-reduce the per-op traffic is $2\frac{n-1}{n}BHb_{\text{act}}$ bytes, and per-op time is bandwidth plus a latency term that dominates at small batch:

$$t_{\text{AR}} = N_{\text{AR}}\left(\frac{2\frac{n-1}{n}BHb_{\text{act}}}{B_{\text{net}}} + \ell\right).$$

EP with DP attention replaces all-reduces with dispatch/combine all-to-alls per MoE layer, with traffic $\approx BL_{\text{MoE}}\,rH(b_{\text{disp}} + b_{\text{comb}})$, where $r = n\big(1 - (1 - 1/n)^k\big)$ is the expected number of distinct expert ranks

touched by a token's top-k selection ($r \approx 6.5$ at k=8, n=16)—the rank-level sibling of the expert-union expectation of §3.3. Dispatch sends one activation per touched rank; combine returns one partial sum per touched rank (an upper bound if the implementation aggregates in-network). The implementation constant is decisive: a low-latency RDMA all-to-all approaches the floor while a naive one can be several times slower. Floor First therefore separates two questions that the literature usually merges: *which route has the better ceiling*, and *does a near-ceiling implementation of that route exist*? Comparisons are made at the ceiling level first; only routes that pass are checked for mature implementations (e.g., DeepEP-class all-to-all), and routes must never be eliminated by benchmarking a poor implementation of them.

3.5 Capacity Wall

The KV capacity wall bounds concurrency:

$$B_{\max} = \frac{(C_{\text{HBM}} - C_{\text{weights}} - C_{\text{overhead}}) \cdot s_{\text{KV}}}{S \cdot \text{bytes}_{\text{KV/token}}},$$

where the KV sharding factor s_{KV} is 1 for MLA under head-parallel TP (full replication), $\min(n, n_{\text{kv}})$ for GQA under TP, and n for DP attention. Because decode goodput is B/t_{step} under a TPOT SLO, a layout with slightly worse same-batch t_{step} but an order-of-magnitude larger $B_{\max}$ can dominate the goodput frontier. Treating capacity as a fifth resource dimension—rather than an afterthought OOM check—is what turns the account into a deployment judgment.

3.6 Compositionality: New Architectures Enter by One Module

The account is modular. Each layer type—attention, routed FFN, dense FFN, or a future state-space block—declares five things: weight bytes, state bytes, per-step state reads, FLOPs, and communication. These quantities are functions of workload (B, S) and of that module's sharding plan. A model is then just a list of modules, and a step account is the per-resource sum over that list. This design has two consequences. First, adding an architecture does not require a new framework. DSA changes the attention module's state-read function from $O(S)$ to approximately $O(k)$ per query (§5.3); a state-space layer would declare a constant-size recurrent state instead of a growing KV cache, and the capacity wall of §3.5 would update automatically. Second, shardability belongs to the module. MLA's latent KV declares itself unshardable across attention heads, so DP attention follows naturally for high-concurrency serving; architecture work that re-enables TP for latent attention [25, 34] is, in this language, changing that declaration. Parallelism plans are therefore per-module assignments rather than global labels. A mixed plan such as TP attention with EP experts—deployed in production [24]—is simply another point in the same space. For standard families, the declarations are mostly data entry: the constants come from published model dimensions, as in the artifact's specification file for dense, GQA, MLA, and MoE models. The current artifact hard-codes the two layouts in the case study; the per-module plan interface is staged as the next revision (§7).

4 Artifact

The artifact is deliberately small: a specification module (`specs.py`) holding model, GPU, and cluster constants with datasheet and calibrated rates kept separate; a floor calculator (`floor.py`) producing the resource table, [max, sum] floors, capacity wall, single-stream bound, and prefill floor; a reconciliation tool (`mbu_mfu.py`) converting measured TPOT/TTFT into MBU/MFU with a triage verdict; and an agent-readable workflow document (`SKILL.md`). The implementation is pure Python with no runtime dependencies; Appendix A gives the commands used to reproduce the case-study tables.

Calibration is outside the model by design. The chain is: HBM bandwidth via `nvbandwidth`; GEMM rates via the production serving shapes; communication via `nccl-tests` and the production all-to-all's own benchmark. Datasheet floors prune the design space; calibrated floors make residuals meaningful.

We executed this chain on the target cluster (fallback tools where installation was constrained: a torch-level device copy for HBM, production DeepGEMM kernels for FP8). GPU-side rates landed near expectations (81% of datasheet HBM bandwidth under the fallback tool, 89% of FP8 peak), but calibration moved

Table 1: Datasheet portrait (Figure 2 renders the geometry). The H20 trades compute for nothing else; its ridge point is 8× lower than H100's, widening the memory-bound comfort zone by the same factor.

	HBM	BW	FP8 dense	Ridge (FLOP/B)
H100 SXM	80 GB	3.35 TB/s	1979 T	∼590
H800	80 GB	3.35 TB/s	1979 T	∼590
H20	96 GB	4.00 TB/s	296 T	∼74

every communication constant: the effective all-reduce rate at decode message sizes is 43 GB/s against a 100 GB/s per-node line rate; small-message latency is ≈33 μs and equal within noise across and within nodes, indicting NCCL launch overhead rather than the wire; and DeepEP's op-level dispatch latency is 60 μs against a ∼30 μs pure-transport bound. A raw-NCCL all-to-all on the same fabric measured 12.6 GB/s where DeepEP approaches line rate—a concrete example of the ceiling-versus-implementation distinction in §3.4. One accounting rule emerged: the bandwidth constant must be the latency-*excluded* slope, not any single message size's effective bus bandwidth, or the latency term is double-counted. Fully calibrated floors are 15–24% looser than datasheet ones and serve as the reconciliation baseline for §8. Notably, although calibration moved every communication constant—one by 8×—no optimistic floor and no deployment judgment of §5 changed: on this hardware the HBM account and the capacity wall carry the verdicts, so they are insensitive to precisely the inputs that are hardest to know in advance. This robustness is a property of wall ordering, not luck: a judgment carried by the first wall survives errors in the accounts of walls never reached.

The `SKILL.md` component addresses a distinct failure mode: coding agents, like humans, reflexively reach for benchmarks and profilers. Encoding the workflow as an agent skill makes the floor-first discipline enforceable in agentic optimization loops [14, 22], where the artifact serves as the deterministic arithmetic layer beneath model-driven judgment: a triggered agent must produce the floor account before proposing any benchmark, and profiler use must cite a residual above threshold. This is not speculative: the calibration campaign of §8-(1) was executed on the production cluster by a coding agent operating under this skill—following the measurement protocol, hitting the documented fallback paths when tooling was unavailable, and returning the spec-keyed calibration file this paper consumes. The artifact, calibration data, and measurement scripts will be released with the next revision of this draft.

5 Case Study: DeepSeek-V3.2-Style MoE on 16×H20

5.1 Hardware Portrait: the H20 Ridge Point

The H20 is the export-compliant Hopper variant: compute is cut to ∼15% of H100 while memory bandwidth is comparable or higher (Table 1). The consequence is a ridge point of 296/4.0 ≈ 74 FLOP/byte versus ≈ 590 for H100: *the H20's memory-bound region is* 8× *wider*. A workload below 74 FLOP/byte is memory-bound on H20, so the compute cut is mostly hidden. Above 74, H20 starts to pay for its weaker tensor throughput; beyond the H100 ridge, both GPUs are compute-bound and the full compute gap appears. Decode at realistic batch sizes sits below the H20 ridge, while prefill sits far above it—so the H20 is naturally suited to decode and weak for prefill. This single number reframes deployment: policy analyses observed empirically that H20 inference throughput rivals flagship parts [24], and kernel work targets its weak tensor throughput [13], but to our knowledge no published account derives the deployment consequences from the ridge geometry.

5.2 Setup

We analyze a DeepSeek-V3.2-style model—671B total / 37B activated parameters, FP8 weights, MLA latent KV (70 KB per token across 61 layers), 58 MoE layers with 256 routed experts and top-8 routing—following the public V3-family dimensions [10, 45] and the DSA sparse-attention direction of V3.2 [12]. Deployment: 2 nodes × 8 H20, 4×200 Gb HDR InfiniBand per node (≈100 GB/s unidirectional aggregate). Communication constants are calibrated on this cluster (§4); GPU-side rates in Table 2 stay at datasheet values

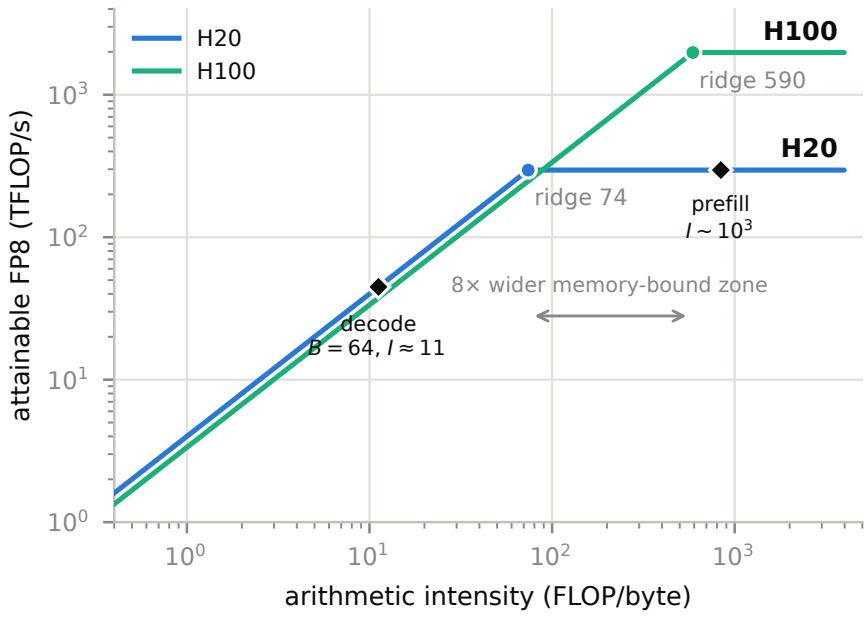

Figure 2: FP8 rooflines of H20 and H100 with the case study's operating points. Decode at B=64 ($I \approx 11$ for the TP16 account) sits below the H20 ridge, so it is governed by memory traffic rather than tensor peak. Prefill ($I \sim 10^3$) sits past both ridges, where the H20 pays its compute cut. The 8× ridge gap is the geometric content of "a decode part."

Table 2: Analytical decode floors, DeepSeek-V3.2-style MoE on 16×H20, B=64, S=8192 (ms per step; conservative expert union; GPU rates at datasheet, collective rates cluster-calibrated—no meaningful datasheet exists for effective collective bandwidth). The union expectation (§3.3) tightens weight terms by ∼13% at this batch.

Configuration	Weight	KV	HBM Σ	Compute	Network	Floor [max, sum]	$B_{\max}$
TP16	10.48	9.21	19.70	2.99	8.91	[19.7, 31.6]	70
TP16 + DSA-style sparse	10.48	2.30	12.79	1.50	8.91	[12.8, 23.2]	70
EP16 + DP attention	14.70	0.58	15.28	2.99	10.66	[15.3, 28.9]	644

for reproducibility, with fully calibrated floors 15–24% looser (§4). The workload is steady-state decode at B=64, S=8192. Table 2 gives the floors; all values reproduce from the artifact.

5.3 Wall Ordering

The account in Table 2 is best read not row-by-row but as an ordering of the walls a deployment hits as load grows (Table 3).

Two structural facts fall out. First, the compute wall is not the next wall. If we compare only parameter GEMMs with full-expert weight reads, the dense rule of thumb $2B \geq I_{\text{ridge}}$ would put the knee at $B \approx 37$. For MoE decode, however, each full weight read supports only the activated fraction of the model, so the GEMM-only knee shifts by the sparsity ratio 671/37 ≈ 18× to $B^*_{\text{GEMM}} \approx 670$. Including the 8K MLA attention FLOPs lowers the compute-vs-weight knee to roughly $B \approx 225$ (roughly 450 with DSA), but this still lies beyond the TP16 capacity wall at $B \approx 70$. Once KV bytes are included, the HBM account grows at least as fast as the compute account, so the feasible TP16 region ends before compute can bind. Chasing decode MFU here is optimizing a wall the deployment cannot reach. Second, walls are removed in order or not at all: DSA removes the KV-bandwidth wall (row 2) but not the capacity wall (row 1), because top-k selection reduces *reads*, not *residency*—the full KV must remain addressable for future steps.

Table 3: Wall ordering for TP16 at 8K context. Each wall is quantified by the floor account; each removal reshuffles the remaining order. The compute wall is unreachable in the feasible region: the capacity wall caps B at 70, before even the attention-aware compute knee (roughly $B \approx 225$ at 8K, and higher after DSA). The GEMM-only MoE-corrected knee is $B^* \approx 670$.

#	Wall	Binds when	Removed by
1	KV capacity	$B \gtrsim 70$	DP attention ($\rightarrow$644)
2	KV bandwidth	long S (9.2 ms @8K)	DSA ($\rightarrow$2.3 ms)
3	Weight bandwidth	$B \gtrsim 32$ (union sat.)	quantization; EP scale-out
4	Compute	$B^*_{\text{GEMM}} \approx 670$; incl. 8K MLA $\sim$225	*capacity comes first*

(The account models DSA's dominant effect, the KV-read reduction. Its lightning indexer still has an $O(S)$ scoring path; we treat it as second-order at 8K but expose it as a module that must be added for long-context accounting [12, 48].) After DSA, the exposed wall is routed-weight bandwidth, which is why quantization and EP-style weight sharding, not more attention work, are the next levers on this hardware.

5.4 TP16: Both KV Terms Replicate

TP16 shards all weights 16 ways ($\sim$42 GB/GPU at full expert union) but MLA replicates KV: each GPU reads the *entire* 36.8 GB of active KV per step (§3.3). The HBM account is 19.7 ms—weight and KV reads add on the same resource—dominating compute (3.0 ms, of which 9.4 of the 14.2 TFLOP is per-layer MLA attention) and network (8.9 ms: 4.9 ms of traffic at the calibrated 43 GB/s effective all-reduce rate, plus 4.0 ms of latency, $122 \times 33\,\mu$s). The latency constant is itself a calibration finding: cross-node and intra-node small-message latency measure equal within noise ($\approx$33 vs. 32.7 μs), identifying ungraphed NCCL launch overhead—not wire time—as the per-op cost, with CUDA-graphed production paths reaching $\sim$10 μs. Sparse attention cuts the KV term to 2.3 ms, but the capacity wall stays at $B_{\max} \approx 70$ for 8K contexts: DSA changes what is read, not what is stored.

5.5 EP16 + DP Attention: Trading Bandwidth for Capacity

EP16 shards routed experts but replicates the non-routed $\sim$18 GB on every GPU, raising per-GPU weight traffic to 58.8 GB (14.7 ms)—worse than TP16's 10.5 ms at equal batch. DP attention, however, shards both KV *reads* (0.58 ms) and KV *residency*: the capacity wall moves from 70 to ≈ 644 concurrent 8K requests, an order of magnitude. Under a TPOT SLO, goodput is B/t_{step} along the feasible region; the layout with the higher wall wins the frontier even where its same-batch step time is worse. This is the quantitative content of the deployment folklore "attention should go DP for MLA serving" [11, 31]: the judgment follows from rows 1–3 of the wall table, needs no benchmark to reach, and—as the next subsection shows—reverses at low concurrency.

5.6 Single-Stream Bounds: the Judgment Inverts

At B=1 the layer-serial chain removes overlap opportunity, so the honest floor is the sum (§3.1). For TP16: weight reads shrink to the touched-expert union ($\sim$2.4 GB, 0.6 ms) but 122 all-reduce latencies cost 4.0 ms at the measured 33 μs/op—latency, not bandwidth, is the single-stream wall—giving TPOT $\geq$ 4.9 ms, i.e. $\lesssim$ 205 tok/s. For EP16+DP: at B=1, the active attention rank reads the replicated non-routed block plus the touched routed experts ($\sim$19.3 GB total, 4.8–4.9 ms), and the step still pays 116 all-to-all latencies at the measured 60 μs (7.0 ms). The resulting bound is TPOT $\geq$ 11.8 ms, or $\lesssim$ 85 tok/s—*TP wins the single-stream regime by* 2.4$\times$, the mirror image of the high-concurrency judgment. Adding GPUs helps neither: because the calibration shows the 33 μs is launch-dominated (§5), the levers are CUDA-graph launch elimination, fewer inter-layer synchronizations, or a faster interconnect—in that order of accessibility.

Two framing notes. These are physical bounds, not achievable expectations: recent measurements show batch-1 decode on high-bandwidth parts reaching only $\sim$27% of its analytic floor due to kernel-launch overheads [7], so single-stream floors bound feasibility studies ("can this SLO ever be met?") rather than predict

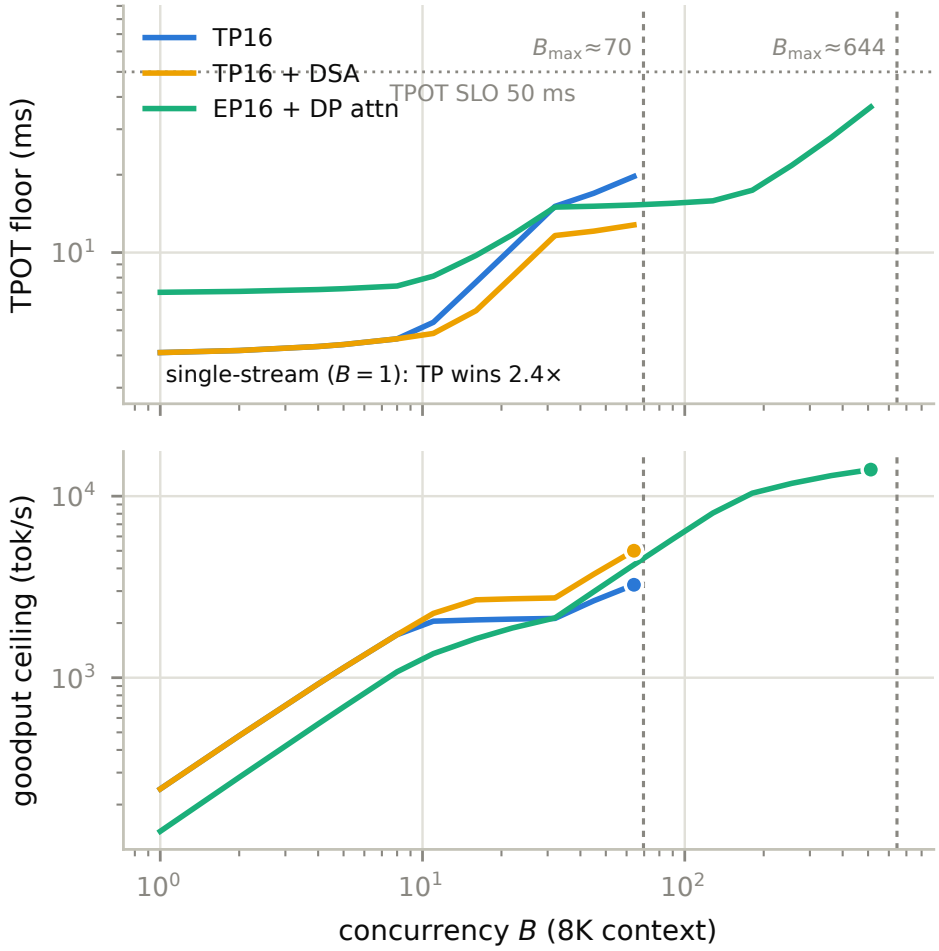

Figure 3: Analytical TPOT floors (top) and goodput ceilings (bottom) versus concurrency at 8K context, computed by the artifact. The KV-capacity walls (dashed) terminate each layout's feasible region: TP16 variants end at $B \approx 70$ while EP16+DP attention continues to $B \approx 644$, dominating the high-concurrency frontier—yet at B=1 the ordering inverts and TP16 leads by nearly 2×. The deployment judgment is a function of the operating point.

production numbers. And the regime dependence of the TP-vs-DP judgment—TP at low concurrency, DP at high—is exactly the shape of the production disagreement in §1: the LMSYS–Ant H20 deployment (TP16 attention) and the official DeepSeek deployment (DP attention) sit at different operating points on the same frontier. The account does not say one team erred; it says the choice is a computable function of the target concurrency, and computes it.

5.7 Reconciliation: Reading Residuals

Reconciliation compares floors against steady-state *service time* (P50 TPOT), not tail latency: queueing delay is real but lives outside the resource account (§7), so P99 validates the SLO while P50 diagnoses the engine—conflating them reads tail queueing as kernel residual.

Suppose TP16 (no sparse attention) measures 25 ms P50 TPOT at this workload. The artifact reports MBU $= (41.9 + 36.8)\,\text{GB}/(25\,\text{ms} \times 4.0\,\text{TB/s}) = 78.8\%$; the residual is 1.27× the optimistic floor—below the escalation threshold—and the position inside $[19.7, 31.6]$ is 0.45. The two signals say different things, which is why both exist: the residual says *stop*—no profiler session is warranted, and the next win must come from

changing the account itself (DSA, quantization, or layout)—while the mid-interval position bounds what better overlap could ever recover (∼5 ms here) if that axis were pursued anyway. Had the measurement been 45 ms (MBU = 44%, 1.42× the *pessimistic* floor), it would sit outside the interval entirely: no overlap explanation is arithmetically possible, and the tool escalates to Nsight Systems with the three-question reading list, because something outside the resource account (host gaps, stragglers, preemption) is consuming time.

For prefill, the identity of §2.2 selects MFU: an 8K prompt costs $2 \cdot 37\text{B} \cdot 8192 \approx 606$ TFLOP of parameter GEMMs, so 16 H20s at 50% MFU bound TTFT at 256 ms—a GEMM-only lower bound (the causal-attention term adds up to ∼25 ms at this length); H100s would bound the same GEMM account at ∼38 ms, the prefill face of the ridge portrait. A measured 400 ms implies 32% MFU, within the MoE prefill band (§2.4); the timeline-level suspects (all-to-all exposure, expert imbalance) are checked before any kernel is blamed.

6 Related Work

Analytical models and limit studies. Pope et al. [27] founded the genre: closed-form partitioning selection for dense Transformers on TPUs. GenZ [4] maps use cases to platform requirements with a resource decomposition close to ours; LLMCompass [44] and Calculon-style co-design models [20] serve hardware DSE; roofline treatments cover the single-device [41] and edge [5, 6] settings. Closest to our decode account, LIMINAL [9] derives single-user decode limits from bandwidth, compute, capacity, and synchronization, and Erdil [15] derives token-economics frontiers including a speed $\propto \text{BW}^{1/3}$ scaling; both fix overlap assumptions and target hardware-evolution questions, where Floor First keeps overlap observable via the $[\max, \text{sum}]$ interval and targets deployment triage. Yun et al. [42] analyze MLA/MoE arithmetic intensity at the operator level, concluding MLA core-attention trends compute-bound; this is consistent with our step-level account—operator intensity and step-level byte totals answer different questions, and after DSA the step is weight-bandwidth-dominated regardless of the attention operator's intensity. AIC++ [38] explores the same V3.2 deployment space (including attention–FFN disaggregation, which we do not model) via measured kernel databases plus network simulation on B200; Floor First trades its fidelity for closed forms, interpretable wall ordering, and hardware it has no database for—the two are complementary stages of the same pipeline.

Simulators and configuration search. Vidur [2] and KernelSight-LM [40] predict serving performance from profiled operators; LLM-Pilot [21] learns black-box cost models across GPUs; AIConfigurator [39] searches configurations against a measured-kernel database in seconds. These tools accelerate the grid search; Floor First's position is that the floor account should shrink the grid *before* any search, look-up, or simulation—and, unlike a kernel database, closed forms transfer to hardware and models nobody has profiled yet. When scheduler dynamics or caching dominate, simulators are the correct escalation target, exactly as profilers are for kernel questions.

Serving systems and the DP-attention judgment. Phase splitting and scheduling [1, 26, 28, 46], module-level disaggregation [49], hybrid MoE parallelism [47], and model–system co-design [32] define the layout space our accounts judge. For the MLA replication problem specifically, the engineering record states the mechanism [11, 31, 36] and architecture papers confirm it while responding by redesigning attention or its sharding [25, 34, 43]; reverse-engineering analyses quantify official H800 deployments [16]. None derives the capacity-first judgment for unmodified MLA, and production deployments still disagree [24]—the gap this case study fills. H20-specific work exists at the kernel level [13] and as empirical throughput reports; the ridge-geometry account of §5.1 appears to be new.

MoE metrics and sparsity accounting. MoE-CAP [19] corrects MBU/MFU for per-token activated parameters; our union correction (§3.3) additionally captures batch-aggregated weight traffic, using a closed form shared with speculative-decoding and expert-sharing analyses [18, 35]. Empirical characterizations [3, 7, 29] document bandwidth saturation, capacity traps, and floor-attainment gaps that our workflow is designed to triage—and that its failure modes (§7) must respect.

Benchmark discipline and agentic optimization. Open-versus-closed-loop bias was formalized by Schroeder et al. [30]; LLM serving evaluation work has begun rebuilding measurement discipline, and §2.1 applies the classical distinction to serving capacity claims. Agentic optimization systems increasingly wrap profilers and roofline reasoning in LLM agents [14, 17, 22, 23, 33]; FLOOR FIRST's skill artifact is the complementary piece—a deterministic floor layer that disciplines when such agents may escalate to expensive tools.

7 Discussion

Why not always simulate? Because many bad deployments can be ruled out with one page of arithmetic. If TP16 cannot hold the required concurrency at the target context, no scheduler search fixes that. If a benchmark already sits at 1.27× the optimistic floor, kernel work cannot return a large factor. Simulators and profilers remain important, but they are stage-2 tools for residuals that survive the floor check.

What generalizes. The mental model is not specific to this case study. The five-dimensional account, the two-sided floor, wall ordering, and the ridge-identity metric discipline apply to any autoregressive transformer served on accelerators: hardware enters through five numbers per device (§5.1), the model through per-module declarations (§3.6), and the deployment through per-module sharding. Nothing in the workflow references H20, MLA, or MoE specifically—the case study was chosen because it exercises the account's hardest current instance (extreme ridge asymmetry, unshardable latent KV, batch-dependent expert traffic), not because it marks a boundary. What does *not* transfer automatically: calibration constants (per cluster), the module library (per architecture family), and the blind spots below. Training-side estimation follows the same resource-vector structure but is out of scope here.

When the floor model fails. The account has known blind spots, and stating them is part of the method. (i) *Continuous batching*: B varies within a window, so floors should bracket the batch distribution rather than assume a stationary point. (ii) *Expert imbalance*: the union expectation assumes uniform routing; hot experts push per-GPU weight traffic above the account (EPLB-style balancing restores it), and the sum floor loses meaning when stragglers, not resources, set step time. (iii) *Open-loop overload*: past saturation, sojourn time is queueing-dominated; floors bound service time only, and queueing-theoretic capacity models take over. (iv) *Host-bound regimes*: Python and scheduler overheads are outside the resource vector by construction—the $< 40\%$ MBU band plus kernel-gap inspection exists precisely to catch what the account cannot express. (v) *Batch-1 attainment*: launch overheads leave measured single-stream latency far above the floor on high-end parts [7]; single-stream floors answer feasibility, not prediction. (vi) *Calibration drift*: per-shape GEMM efficiency varies widely; the calibration chain must use production shapes or residuals inherit the error.

Scope. The artifact currently models pure TP and EP+DP-attention decode plus compute-bound prefill. The compositional structure of §3.6 defines the extension path: a per-module plan interface admits mixed TP×EP and the production hybrid layouts of §1; new attention or state-space modules enter by declaring their state and shardability; prefill/decode disaggregation, chunked-prefill interference, speculative decoding, and attention–FFN disaggregation [38] are account extensions on the same interface. All are staged behind validation of the core loop—the framework does not change, the module list grows.

8 Required Measurement Campaign

The workflow's claims are calibratable, and this section is the paper's own stage-2 escalation plan. (1) *Calibration chain* on the target cluster—*executed* (§4): production-shape GEMM rates, all-reduce bandwidth/latency sweeps, DeepEP all-to-all benchmarks; the HBM point should be re-taken with `nvbandwidth` in place of the torch fallback. (2) *Residual calibration*: an open-loop, steady-state sweep over {TP16, TP16+DSA, EP16+DPA} × B × S ($\geq$12 points), each reconciled against its floor, yielding the residual distribution that grounds (or revises) the 1.3× threshold and the MBU bands. (3) *Triage traces*: one near-floor and one

high-residual configuration instrumented with Nsight Systems, demonstrating the three-question reading list and the budget-table indictment of a kernel class. (4) *Cross-platform replication* of the workflow on a dense model and a second GPU generation, testing that the method—not the case study—is the contribution. (5) *Judgment validation*: the TP16-attention versus DP-attention goodput frontiers measured on the same 16×H20 system, closing the loop on §5.6's claim that the production disagreement is an operating-point question.

9 Conclusion

Floor First reframes LLM serving optimization as residual-driven triage: compute two-sided floors from a five-dimensional resource account, benchmark only the configurations whose walls permit the target, read the $[\max, \text{sum}]$ position as a free overlap diagnostic, and open profilers only when the residual points to time the account cannot explain. The H20 case study shows the method's reach: a ridge-point portrait that explains why an export-constrained part is a natural decode engine, a wall ordering that shows its compute ceiling is unreachable behind the KV-capacity wall, a capacity-first derivation of the DP-attention judgment together with its single-stream inversion—and, throughout, arithmetic that fits on a page and reproduces from a dependency-free artifact. The measurement campaign of §8 is the remaining step from analytical draft to validated systems paper.

References

[1] Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. *arXiv preprint arXiv:2403.02310*, 2024.

[2] Amey Agrawal et al. Vidur: A large-scale simulation framework for llm inference. *arXiv preprint arXiv:2405.05465*, 2024.

[3] Moiz Arif, Avinash Maurya, Sudharshan Vazhkudai, and Bogdan Nicolae. Understanding inference scaling for llms: Bottlenecks, trade-offs, and performance principles. *arXiv preprint arXiv:2605.19775*, 2026.

[4] Abhimanyu Bambhaniya, Ritik Raj, Geonhwa Jeong, Souvik Kundu, Sudarshan Srinivasan, Suvinay Subramanian, Midhilesh Elavazhagan, Madhu Kumar, and Tushar Krishna. Demystifying ai platform design for distributed inference of next-generation llm models. *arXiv preprint arXiv:2406.01698*, 2024.

[5] Zhen Bi, Xueshu Chen, Luoyang Sun, Yuhang Yao, Qing Shen, Jungang Lou, and Cheng Deng. Rooflinebench: A benchmarking framework for on-device llms via roofline analysis. *arXiv preprint arXiv:2602.11506*, 2026.

[6] Hao Chen, Cong Tian, Zixuan He, Bin Yu, Yepang Liu, and Jialun Cao. Inference performance evaluation for llms on edge devices with a novel benchmarking framework and metric. *arXiv preprint arXiv:2508.11269*, 2025.

[7] Josef Chen. Memory-bound but not bandwidth-limited: The physical ai inference gap in batch-1 llm decode. *arXiv preprint arXiv:2605.30571*, 2026.

[8] Databricks Engineering. Llm inference performance engineering: Best practices. https://www.databricks.com/blog/llm-inference-performance-engineering-best-practices, 2023. Introduces the model bandwidth utilization (MBU) metric.

[9] Michael Davies, Neal Crago, Karthikeyan Sankaralingam, and Christos Kozyrakis. Liminal: Exploring the frontiers of llm decode performance. *arXiv preprint arXiv:2507.14397*, 2025.

[10] DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[11] DeepSeek-AI. Deepseek-v3/r1 inference system overview and profiling data. https://github.com/deepseek-ai/profile-data, 2025. Production deployment: prefill EP32, decode EP144 with data-parallel attention on H800.

[12] DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[13] Pengcuo Dege, Qiuming Luo, Rui Mao, and Chang Kong. Flashmla-etap: Efficient transpose attention pipeline for accelerating mla inference on nvidia h20 gpus. *arXiv preprint arXiv:2506.01969*, 2025.

[14] Yuran Ding, Xinwei Chen, Xiaofan Zhang, and Zongwei Zhou. Asap: an agentic solution to auto-optimize performance of large-scale llm training. *arXiv preprint arXiv:2511.03844*, 2025.

[15] Ege Erdil. Inference economics of language models. *arXiv preprint arXiv:2506.04645*, 2025.

[16] Jiarui Fang. Deepseek v3/r1 inference efficiency analysis (in chinese). https://zhuanlan.zhihu.com/p/16445683081, 2025. Reverse-engineered roofline account of the official H800 deployment; companion tool LLMRoofline.

[17] Jiading Gai, Shuai Zhang, Kaj Bostrom, Jin Huang, Vihang Patil, Haoyang Fang, Bernie Wang, Huzefa Rangwala, and George Karypis. Optimizing cuda like a human: Micro-profiling tools as expert surrogates for llm-based gpu kernel optimization. *arXiv preprint arXiv:2606.26453*, 2026.

[18] Zongle Huang, Lei Zhu, Zongyuan Zhan, Ting Hu, Weikai Mao, Xianzhi Yu, Yongpan Liu, and Tianyu Zhang. Moesd: Unveil speculative decoding's potential for accelerating sparse moe. *arXiv preprint arXiv:2505.19645*, 2025.

[19] Yinsicheng Jiang, Yao Fu, Yeqi Huang, Ping Nie, Zhan Lu, Leyang Xue, Congjie He, Man-Kit Sit, Jilong Xue, Li Dong, et al. Moe-cap: Benchmarking cost, accuracy and performance of sparse mixture-of-experts systems. *arXiv preprint arXiv:2412.07067*, 2024.

[20] Joyjit Kundu, Wenzhe Guo, Ali BanaGozar, Udari De Alwis, Sourav Sengupta, Puneet Gupta, and Arindam Mallik. Performance modeling and workload analysis of distributed large language model training and inference. *arXiv preprint arXiv:2407.14645*, 2024.

[21] Małgorzata Łazuka, Andreea Anghel, and Thomas Parnell. Llm-pilot: Characterize and optimize performance of your llm inference services. *arXiv preprint arXiv:2410.02425*, 2024.

[22] Jiajie Li, Erwei Wang, Zhiru Zhang, and Samuel Bayliss. From human guidance to autonomy: Agent skill system for end-to-end llm deployment on spatial npus. *arXiv preprint arXiv:2606.07586*, 2026.

[23] Zhanlin Liu, Yitao Li, and Munirathnam Srikanth. Epoch: An agentic protocol for multi-round system optimization. *arXiv preprint arXiv:2603.09049*, 2026.

[24] LMSYS Org and Ant Group. Serving deepseek-r1 on h20-96g. https://www.lmsys.org/blog/2025-09-26-sglang-ant-group/, 2025. Decode uses TP16 attention with EP16 MoE on 2×8 H20-96G.

[25] Fanxu Meng. Gqla: Group-query latent attention for hardware-adaptive large language model decoding. *arXiv preprint arXiv:2605.15250*, 2026.

[26] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Inigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. *arXiv preprint arXiv:2311.18677*, 2023.

[27] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *arXiv preprint arXiv:2211.05102*, 2022.

[28] Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A kvcache-centric disaggregated architecture for llm serving. *arXiv preprint arXiv:2407.00079*, 2024.

[29] Pol G. Recasens, Ferran Agullo, Yue Zhu, Chen Wang, Eun Kyung Lee, Olivier Tardieu, Jordi Torres, and Josep Ll. Berral. Mind the memory gap: Unveiling gpu bottlenecks in large-batch llm inference. *arXiv preprint arXiv:2503.08311*, 2025.

[30] Bianca Schroeder, Adam Wierman, and Mor Harchol-Balter. Open versus closed: A cautionary tale. In *3rd Symposium on Networked Systems Design and Implementation (NSDI 06)*, 2006.

[31] SGLang Team. Sglang v0.4: Zero-overhead batch scheduler, cache-aware load balancer, faster structured outputs. https://lmsys.org/blog/2024-12-04-sglang-v0-4/, 2024. Introduces data-parallel attention for MLA: head-parallel TP duplicates the latent KV cache.

[32] StepFun. Step-3 is large yet affordable: Model-system co-design for cost-effective decoding. *arXiv preprint arXiv:2507.19427*, 2025.

[33] Qitong Sun, Jun Han, Tianlin Li, Zhe Tang, Sheng Chen, Fei Yang, Aishan Liu, Xianglong Liu, and Yang Liu. Kernelskill: A multi-agent framework for gpu kernel optimization. *arXiv preprint arXiv:2603.10085*, 2026.

[34] Xiaojuan Tang, Fanxu Meng, Pingzhi Tang, Yuxuan Wang, Di Yin, Xing Sun, and Muhan Zhang. Tpla: Tensor parallel latent attention for efficient disaggregated prefill and decode inference. *arXiv preprint arXiv:2508.15881*, 2025.

[35] Daniil Vankov, Nikita Ivkin, Kyle Ulrich, Xiang Song, Ashish Khetan, and George Karypis. Xshare: Collaborative in-batch expert sharing for faster moe inference. *arXiv preprint arXiv:2602.07265*, 2026.

[36] vLLM Project. RFC: Data parallel attention and expert parallel moes. https://github.com/vllm-project/vllm/issues/16037, 2025.

[37] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009.

[38] Hanjiang Wu, Abhimanyu Rajeshkumar Bambhaniya, Sarbartha Banerjee, Tuhin Khare, Sudarshan Srinivasan, Suvinay Subramanian, Souvik Kundu, Madhu Kumar, Midhilesh Elavazhagan, William Won, Amir Yazdanbakhsh, and Tushar Krishna. How far can disaggregation go? a design-space exploration of attention-ffn disaggregation for efficient moe llm serving. *arXiv preprint arXiv:2605.28302*, 2026.

[39] Tianhao Xu, Yiming Liu, Xianglong Lu, Yijia Zhao, Xuting Zhou, Aichen Feng, Yiyi Chen, Yi Shen, Qin Zhou, Xumeng Chen, et al. Aiconfigurator: Lightning-fast configuration optimization for multi-framework llm serving. *arXiv preprint arXiv:2601.06288*, 2026.

[40] Xiteng Yao, Taeho Kim, Hengzhi Pei, Xinle Liu, Kyle Ulrich, Leonard Lausen, Ashish Khetan, Xiang Song, George Karypis, and Martin Herbordt. Kernelsight-lm: A kernel-level llm inference simulator. *arXiv preprint arXiv:2606.28565*, 2026.

[41] Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, et al. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*, 2024.

[42] Sungmin Yun, Seonyong Park, Hwayong Nam, Younjoo Lee, Gunjun Lee, Kwanhee Kyung, Sangpyo Kim, Nam Sung Kim, Jongmin Kim, Hyungyo Kim, et al. Rethinking llm inference bottlenecks: Insights from latent attention and mixture-of-experts. *arXiv preprint arXiv:2507.15465*, 2025.

[43] Ted Zadouri, Hubert Strauss, and Tri Dao. Hardware-efficient attention for fast decoding. *arXiv preprint arXiv:2505.21487*, 2025.

[44] Hengrui Zhang, August Ning, Rohan Prabhakar, and David Wentzlaff. A hardware evaluation framework for large language model inference. *arXiv preprint arXiv:2312.03134*, 2023.

[45] Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Huazuo Gao, Jiashi Li, Liyue Zhang, Panpan Huang, Shangyan Zhou, Shirong Ma, et al. Insights into deepseek-v3: Scaling challenges and reflections on hardware for ai architectures. *arXiv preprint arXiv:2505.09343*, 2025.

[46] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. Distserve: Disaggregating prefill and decoding for goodput-optimized large language model serving. *arXiv preprint arXiv:2401.09670*, 2024.

[47] Bowen Zhou, Jinrui Jia, Wenhao He, Yong Zhang, and Fang Dong. Mixserve: An automatic distributed serving system for moe models with hybrid parallelism based on fused communication algorithm. *arXiv preprint arXiv:2601.08800*, 2026.

[48] Ruijie Zhou, Fanxu Meng, Yufei Xu, Tongxuan Liu, Guangming Lu, Muhan Zhang, and Wenjie Pei. Misa: Mixture of indexer sparse attention for long-context llm inference. *arXiv preprint arXiv:2605.07363*, 2026.

[49] Ruidong Zhu, Ziheng Jiang, Chao Jin, Peng Wu, Cesar A. Stuardo, Dongyang Wang, Xinlei Zhang, Huaping Zhou, Haoran Wei, Yang Cheng, et al. Megascale-infer: Serving mixture-of-experts at scale with disaggregated expert parallelism. *arXiv preprint arXiv:2504.02263*, 2025.

A Artifact Commands

The following commands reproduce Table 2 (add `--full-experts` for the conservative union bound shown; omit it for the expectation account):

```
cd scripts/
python3 floor.py --model deepseek-v3.2 --gpu h20 \
  --nodes 2 --gpus-per-node 8 --parallel tp16 \
  -B 64 -S 8192 --full-experts

python3 floor.py --model deepseek-v3.2 --gpu h20 \
  --nodes 2 --gpus-per-node 8 --parallel tp16 \
  -B 64 -S 8192 --full-experts --dsa

python3 floor.py --model deepseek-v3.2 --gpu h20 \
  --nodes 2 --gpus-per-node 8 --parallel ep16-dpa \
  -B 64 -S 8192 --full-experts
```

Single-stream bounds (§5.6) are printed by the same invocations at `-B 1`; reconciliation examples:

```
python3 mbu_mfu.py decode --model deepseek-v3.2 --gpu h20 \
  --parallel tp16 -B 64 -S 8192 --tpot-ms 25 --full-experts

python3 mbu_mfu.py prefill --model deepseek-v3.2 --gpu h20 \
  --ngpus 16 --prompt 8192 --ttft-ms 400
```