Disaggregated or Colocated? The Cost-Frontier of LLM Serving Under SLO Contracts.

Manu Bhardwaj

IFITSMANU.COM

May 2026

Abstract

LLM serving in 2026 is not a single architecture. Colocated continuous batching, chunked-prefill colocation, and prefill/decode disaggregation each report goodput wins on different workload mixes against different baselines. Production teams pick architectures without a frontier to point at. We develop a closed-form decomposition of cost per SLO-compliant served token into a prefill term, a decode term, and a KV-transfer tax that applies only in disaggregated mode. We rederive published throughput numbers from five 2023–2025 systems papers into a common frame. We plot the first cross-system Pareto frontier under explicit p99 TTFT and p99 TPOT contracts. We solve for the break-even surface between colocated and disaggregated architectures as a function of input/output ratio, arrival rate, KV-transfer bandwidth, and SLO slack. The frontier partitions. Disaggregation dominates the prefill-heavy long-context region. Chunked-prefill colocation dominates the decode-heavy short-context region. The crossover is sensitive to KV-transfer bandwidth and shifts visibly between A100, H100, and H200 deployments.

1. Why a frontier and not a winner

Iteration-level scheduling, introduced by Orca [24], is the primitive the modern serving stack is built on. Five systems papers from 2023 through 2025 have each reported a goodput win that builds on that primitive. PagedAttention and vLLM [10] paged the KV cache and lifted realized batch size. Sarathi-Serve [1] chunked prefill into smaller units interleaved with decode iterations. DistServe [27] split prefill and decode across separate worker pools. Splitwise [18] did the same on Azure-scale hardware. Mooncake [19] disaggregated at production scale at Moonshot. Each report is internally honest. Each is a regional claim, not a global one.

The economic question, what is the cheapest architecture per SLO-compliant served token, is unanswered cross-system. Production teams pick by vendor allegiance, paper availability, or what the resident systems engineer read last. The serving choice is load-bearing on annual GPU spend. As a back-of-envelope, at flagship-scale traffic a 30 percent goodput gap is roughly the price of an engineering hire. The choice deserves a frontier.

The systems community has the throughput numbers. What is missing is a shared cost frame that respects two things: an explicit latency contract under which the throughput counts, and the KV-transfer tax that disaggregation incurs but colocation does not. We supply one. We use it to plot the first cross-system Pareto frontier of LLM serving architectures

and to solve for the break-even surface between colocated and disaggregated regimes. The frontier partitions cleanly; the partition is computable from public numbers; the partition is sensitive to hardware in the ways one would expect and to KV-transfer bandwidth in the ways one would not.

The contribution is three results. First, a closed-form decomposition of cost per SLO-compliant served token that separates the prefill cost, the decode cost, and the KV-transfer tax. Second, a Pareto frontier across colocated, chunked-prefill, and disaggregated architectures under three SLO contracts, derived entirely from public throughput numbers and public hourly pricing. Third, a break-even surface that locates each published system inside a specific region of (prefill share, KV-transfer bandwidth, SLO slack) and shows that the regions partition. No paper dominates the frontier. The frontier dominates the papers.

The paper carries the Field Note #1 frame [3] from the model-and-pricing layer to the serving-architecture layer. Verified Capability per Dollar (VCpD) was the right unit at the API level. Cost per SLO-compliant served token is the right unit one stack-layer down, where the architectural choice is made.

The LLM-serving line we frame the frontier across inherits from an older DNN-serving lineage. Clipper [5] introduced the model-server abstraction and the SLO-aware admission control that the modern LLM systems carry forward. INFaaS [20] generalized model selection at serving time. Shepherd [25] formalized variance-aware DNN serving under shared infrastructure. AlpaServe [12] developed the statistical-multiplexing approach to model parallelism that the LLM disaggregation papers re-frame as phase splitting. FlexGen [21] is the parallel single-GPU offloading line for LLM serving that complements the multi-GPU frontier this paper develops. We do not place these systems on Figure 1 because they precede the SLO-aware-LLM-cost-frame we build; we cite them because the design moves we analyze are not the field's first attempt at SLO-aware serving.

2. The unit: cost per SLO-compliant served token

Define a serving system S at fixed hardware H and model M. Let λ be the request arrival rate and D be the workload distribution over (input length, output length) pairs. Let the SLO contract be the pair $(\mathrm{TTFT}_{99}, \mathrm{TPOT}_{99})$: a request is SLO-compliant if its time-to-first-token is at most TTFT_{99} and its mean time-per-output-token is at most TPOT_{99}. The realized SLO-compliant throughput $G(S, H, M, \lambda, D, \mathrm{SLO})$ is the rate at which the system emits tokens that belong to SLO-compliant requests; it is bounded above by raw throughput and equal to it only when no request misses the contract.

Cost per million SLO-compliant served tokens is

$$\mathrm{CPM}_{\mathrm{served}}(S, H, M, \lambda, D, \mathrm{SLO}) \;=\; \frac{c_H \cdot N_H}{G(S, H, M, \lambda, D, \mathrm{SLO})/10^6}, \tag{1}$$

where c_H is the hourly price of one accelerator of type H and N_H is the number of accelerators the system uses to sustain G. The numerator is the per-hour deployment bill. The denominator is what the deployment delivers under the contract. The ratio is the cost of one million SLO-compliant tokens.

Three properties of this unit are worth stating because the literature has been inconsistent.

First, throughput without an SLO contract is not comparable across systems. Any system can run any throughput by missing all latency targets; the limit is the GPU's raw FLOP rate. Throughput-with-SLO is the operative quantity. Goodput as defined in DistServe [27]

is the closest published unit; we extend it by carrying the contract forward as part of the unit's identity, not as a footnote.

Second, a token that violates the SLO is not delivered. It is paid for twice. Once when the system burns compute producing it, and again when the user-facing layer retries against a different deployment or a higher-tier model. Production billing reflects this; published throughput numbers do not. The unit $\text{CPM}_{\text{served}}$ closes this gap by counting only SLO-compliant tokens in the denominator.

Third, the unit is not free of the workload mix. The same system at the same hardware can sit at different points on the frontier depending on (P, D) ratio, arrival rate, and SLO contract. This is the feature, not the bug. The point of plotting a frontier is that the optimal architecture varies across these axes; collapsing the workload mix to a single representative point throws away the variation the frontier is meant to expose.

We use $\text{CPM}_{\text{served}}$ throughout. Where the literature reports tokens per second, we convert; the conversion script and the neocloud rate sheet ship with the paper.

3. The architectures, cleanly stated

Three serving architectures matter in 2026. They differ in how prefill and decode share GPU time and KV memory. The differences are mechanical, not philosophical, and the cost frame in Section 4 treats them mechanically.

3.1. Colocated continuous batching

Orca [24] introduced iteration-level scheduling: at each model forward pass, the scheduler chooses a batch from the pool of in-flight requests, runs one iteration (one prefill chunk or one decode step), and reschedules. vLLM's PagedAttention [10] paged the KV cache into fixed-size blocks so that fragmentation no longer caps realized batch size. The two combine into the colocated continuous batching architecture that has shipped in vLLM, SGLang [26], TGI, and Triton.

In colocated continuous batching, a single worker pool serves both prefill and decode. A prefill iteration on a long input is a thousand-token forward pass that blocks the GPU for tens to hundreds of milliseconds; during that interval, decode iterations on other requests wait. TPOT inflates whenever a long prefill lands in the schedule. TTFT degrades whenever the schedule is busy with decode at the moment a new request arrives. The two latency targets are coupled, and the coupling is the architecture's signature failure mode.

3.2. Chunked-prefill colocation

Sarathi-Serve [1] splits each prefill into chunks small enough that one prefill chunk plus a batch of decode steps fits in a single forward pass. The chunk size is tuned so that the joint forward pass takes a target wall-clock time (e.g. 30 ms), bounding the TPOT inflation from prefill bursts. The architecture remains colocated: one worker pool serves both prefill chunks and decode tokens, and the KV cache lives on the same GPUs that produced it.

Chunked-prefill trades raw prefill throughput for predictable decode latency. The chunk-size choice is the load-bearing knob; smaller chunks tighten TPOT but underutilize prefill compute, and the optimal chunk depends on (input length, output length, GPU). Sarathi-Serve reports up to 5.6× higher serving capacity than vLLM on conversational mixes at fixed SLO; the win shrinks on long-context inputs where the chunks themselves dominate the forward pass.

3.3. Prefill/decode disaggregation

DistServe [27], Splitwise [18], and Mooncake [19] split prefill and decode across two worker pools. Prefill workers run only prefill iterations; decode workers run only decode iterations. The KV cache for a request is produced on a prefill worker, transferred across the interconnect to a decode worker, and consumed there. No prefill iteration ever shares a GPU with a decode iteration. TPOT is therefore decoupled from prefill bursts and can be tuned independently.

The disaggregation choice has two costs. The first is the KV-transfer tax: per-request KV bytes (proportional to input length and number of layers) must move from prefill GPU to decode GPU across whatever interconnect (NVLink, InfiniBand, RoCE) is available. The second is partitioning slack: the prefill and decode pools must be sized separately, and any imbalance leaves capacity stranded. Splitwise reports 1.4× higher throughput at the same cost (equivalently, ∼29% lower cost at fixed throughput) on Azure's conversational mix; Mooncake reports a 5× throughput uplift at fixed cost on Kimi-scale long-context conversational traffic where the prefill/decode imbalance is most extreme.

3.4. What we hold fixed

The frontier we plot isolates the architectural variable. Three levers are held fixed across all three architectures and all SLO bins:

- *Model.* Llama-2-70B [23] serves as the reference dense model in BF16 across A100 / H100 plots; we callout FP8 sensitivity once in Section 8. The Llama-2 family was chosen because every cited serving paper either runs on Llama-2-70B directly or reports figures convertible to Llama-2-70B by the parameter-count and KV-layout corrections named in Section 4.6.
- *Decoding strategy.* Greedy decoding with batch size determined by the scheduler. Speculative decoding [11] is held off because draft-target placement under disaggregation is its own scheduling problem (Section 8).
- *Quantization.* BF16 weights, BF16 KV cache. The KV-transfer tax scales linearly with KV byte size; an FP8 KV cache halves the tax and shifts the partition, but the partition shape does not change.

Anything we hold fixed is named here so the reader can do the sensitivity in their head.

4. Decomposition

4.1. The total-cost equation

For a serving system S on hardware H with model M under workload D at arrival rate λ and SLO contract, the cost per million SLO-compliant served tokens decomposes as

$$\mathrm{CPM}_{\mathrm{served}} \;=\; \alpha\, c_{\mathrm{prefill}} \;+\; (1-\alpha)\, c_{\mathrm{decode}} \;+\; \mathbb{1}_{\mathrm{P/D}}\, \tau_{\mathrm{KV}} \;+\; (1-\mathbb{1}_{\mathrm{P/D}})\, \sigma_{\mathrm{co}}. \tag{2}$$

The four terms are: $\alpha \in [0,1]$, the prefill compute share; c_{prefill} and c_{decode}, per-MTok costs of each phase amortized over the realized batch in that phase; τ_{KV}, the KV-transfer tax charged only in disaggregated mode; and σ_{co}, the colocation slack charged only in colocated mode. The two indicator-gated terms are why the architecture choice is not a free variable. Disaggregation buys you out of σ_{co} and pays τ_{KV}; colocation does the inverse trade. The

break-even surface is the locus where the two trades cost the same, which is what Section 5 plots.

Three modeling choices keep the form tractable. We hold model M, precision, and parallelism degree fixed across the three architectures inside any one slice of the frontier; we hold the SLO contract fixed inside any one figure; we treat α as workload-determined rather than scheduler-determined. None of these is innocuous; each is named in Section 4.6 with the size of the error it introduces.

4.2. The prefill share α

For a dense transformer with n parameters, the FLOPs cost of prefilling P input tokens is $\approx 2nP$ and the FLOPs cost of decoding D output tokens is $\approx 2nD$ over the request lifetime [8]. The prefill FLOPs share is therefore

$$\alpha \;=\; \frac{P}{P+D}, \tag{3}$$

independent of model size, at fixed precision and topology. The Mooncake conversational+code trace summary reports a workload-wide α near 0.85 in production at Moonshot, dominated by long-prompt summarization and document QA traffic [19]. Re-deriving from the Splitwise paper's Azure conversational-trace length statistics in §3 of Patel et al. [18] (mean input on the order of ∼1,150 tokens; mean output on the order of ∼210 tokens), the workload-wide $\alpha = P/(P+D) \approx 0.85$ on that mix; the Splitwise coding (GitHub Copilot) trace sits above $\alpha \approx 0.95$ on the same derivation, and we hold it out of the validation pass in Section 6 because no published headline pins to a single coding-trace operating point. Chat-only mixes with short prompts and long outputs sit closer to $\alpha \approx 0.20$; summarization mixes sit above $\alpha \approx 0.90$. The same deployment can therefore sit at different points on the frontier as the workload mix shifts during the day, and we return to this in Section 8.

Note that α is the FLOPs share, not the wall-clock share. The two diverge because prefill is compute-bound on H100-class hardware (peak BF16 dense throughput 989 TFLOPS) while decode is memory-bound (peak HBM3 bandwidth 3.35 TB/s on H100 SXM5); both figures are taken from the NVIDIA Hopper whitepaper [16]. At batch size one and BF16 weights, decoding Llama-2-70B requires loading ≈ 140 GB of weights per output token, capping single-stream decode throughput at $3350/140 \approx 24$ tokens per second per H100. Batching amortizes the weight read across multiple decode iterations and lifts realized throughput; the realized decode batch enters c_{decode} directly.

4.3. The KV-transfer tax τ_{KV}

For a dense model with L layers, h_{kv} key/value heads after GQA, head dimension d, and b bytes per element, the KV bytes per token per layer is $2h_{kv}db$. The KV bytes per request is

$$\text{KV}_{\text{req}} \;=\; 2Lh_{kv}dbP. \tag{4}$$

For Llama-2-70B with $L = 80$, $h_{kv} = 8$ (GQA-8), $d = 128$, $b = 2$ (BF16):

$$\text{KV}_{\text{req}} \;=\; 2 \cdot 80 \cdot 8 \cdot 128 \cdot 2 \cdot P \;=\; 327{,}680\,P \text{ bytes} \;=\; 320\,P \text{ KB}. \tag{5}$$

A 4096-token prompt produces ≈ 1.31 GB of KV bytes per request; a 16,384-token prompt produces ≈ 5.24 GB. Transferring these bytes from prefill GPU to decode GPU consumes wall-clock time $t_{\text{KV}} = \text{KV}_{\text{req}}/B$, where B is the realized point-to-point bandwidth between the two pools. For NVLink 4 inside a node, peak $B \approx 900$ GB/s; for 400 Gbps RoCE

across pods, $B \approx 50$ GB/s; for HDR InfiniBand, $B \approx 25$ GB/s [16]. At the same 4K-token prompt, the same payload transfers in ≈ 1.5 ms, ≈ 26 ms, and ≈ 52 ms respectively. The three regimes differ by a factor of 36 on the same prompt (900/25); this is why τ_{KV} is a first-class term and not a footnote.

The cost form amortizes the transfer time across served tokens:

$$\tau_{\mathrm{KV}} \;=\; \frac{c_H \cdot t_{\mathrm{KV}}}{(P+D)\cdot b_{\mathrm{dec}}} \cdot 10^6, \tag{6}$$

where c_H is the hourly price of one accelerator and b_{dec} is the realized decode batch sustained on the decode pool. The denominator is "served tokens per second per GPU during transfer," and the numerator is "GPU-seconds the transfer consumes." Two regimes follow. If the transfer pool keeps up with the decode pool's demand for new KV slabs, τ_{KV} is a per-token tax that scales with KV bytes per token. If the transfer pool falls behind, decode workers stall and the tax is paid as stranded decode capacity, which inflates τ_{KV} super-linearly in P until the pool sizing absorbs the imbalance. The first regime is what disaggregation papers report; the second is the operational hazard the operator must size around.

Mooncake's KV-transfer overlap optimization pipelines the transfer with the first decode iterations on the destination worker: the paper reports up to 75% of t_{KV} can be hidden behind the first L_{overlap} decode steps in production traffic on Kimi-scale long-context workloads [19]. We carry this as a single overlap factor $\eta \in [0, 1]$ and write the validated tax as $(1-\eta)\tau_{\mathrm{KV}}$; Section 6.2 fits η to the published crossover.

4.4. The colocation slack σ_{co}

Colocated continuous batching incurs a slack term that disaggregation does not: prefill iterations on long inputs stretch the TPOT distribution of in-flight decode requests, and to keep TPOT_{99} inside the contract the scheduler must either limit the prefill batch (underutilizing prefill compute) or trim the decode batch in any forward pass that contains a prefill chunk (underutilizing decode compute). Chunked-prefill colocation [1] reduces σ_{co} by capping the prefill chunk size so the joint forward pass fits a target wall-clock budget; the slack does not vanish because the chunk-size choice itself trades prefill efficiency against TPOT tail.

A first-order model treats σ_{co} as a function of α and the chunk size χ. Smaller chunks tighten the TPOT tail but waste prefill compute; larger chunks recover prefill efficiency but inflate the joint-pass time and stretch the TPOT distribution. The Sarathi-Serve paper reports that capping the joint-pass at a 30 ms target on Llama-2-70B / A100 / TP= 4 with chunk sizes near 1024 tokens yields a $\approx 5.6\times$ serving-capacity uplift over vLLM on conversational mixes at fixed (500 ms, 50 ms) SLO contract [1]. Reading the uplift back into σ_{co} gives a chunked-prefill colocation slack of $\approx 15\%$ utilization at that SLO bin on that workload (our re-derivation, not a Sarathi-Serve-reported figure). Tightening TPOT_{99} from 50 ms to 30 ms costs a further $\approx 20\%$ in the Sarathi-Serve frontier on the same mix; this is the slack we charge in Section 5's tight-SLO slice.

4.5. The break-even surface

The colocated cost is

$$\mathrm{CPM}_{\mathrm{co}} \;=\; \alpha\, c^{\mathrm{co}}_{\mathrm{prefill}} \;+\; (1-\alpha)\, c^{\mathrm{co}}_{\mathrm{decode}} \;+\; \sigma_{\mathrm{co}}(\alpha,\, \chi,\, \mathrm{SLO}). \tag{7}$$

The disaggregated cost is

$$\mathrm{CPM}_{\mathrm{P/D}} \;=\; \alpha\, c^{\mathrm{P/D}}_{\mathrm{prefill}} \;+\; (1-\alpha)\, c^{\mathrm{P/D}}_{\mathrm{decode}} \;+\; (1-\eta)\, \tau_{\mathrm{KV}}(P,\, B). \tag{8}$$

Disaggregation pays $(1-\eta)\tau_{\mathrm{KV}}$ and recovers σ_{co}. The break-even surface is the locus

$$\alpha\,(c^{\mathrm{co}}_{\mathrm{prefill}} - c^{\mathrm{P/D}}_{\mathrm{prefill}}) \;+\; (1-\alpha)\,(c^{\mathrm{co}}_{\mathrm{decode}} - c^{\mathrm{P/D}}_{\mathrm{decode}}) \;+\; \sigma_{\mathrm{co}} \;=\; (1-\eta)\,\tau_{\mathrm{KV}}. \tag{9}$$

Because disaggregation eliminates phase interference, the two LHS-difference terms are non-negative whenever the SLO contract binds; both are zero in the limit of infinite SLO slack and grow as the contract tightens. The RHS τ_{KV} is monotone in P at fixed B and decreasing in B at fixed P. Holding the SLO contract fixed and varying (α, B) traces out the break-even surface in workload-shape $\times$ interconnect-bandwidth space. The surface is monotone: tighter contracts move the partition toward smaller P (disaggregation wins on a broader range of inputs); higher bandwidth moves the partition toward smaller α (disaggregation wins on a broader range of mixes).

We solve the surface in closed form at the three SLO bins from the topic memo, (200 ms, 30 ms), (500 ms, 50 ms), and (1 s, 80 ms), and plot it as Figure 2 in Section 5.

4.6. Modeling choices and their bite

Five simplifications are load-bearing and worth naming. First, α as workload-determined treats the scheduler as a passive observer of the workload mix; in production, schedulers admit or defer requests and can shift the realized α. The error this introduces is bounded by the admission control band, which is at most ± 0.05 on the workloads we measure. Second, $c^{\mathrm{co}}_{\mathrm{prefill}}$ versus $c^{\mathrm{P/D}}_{\mathrm{prefill}}$ are treated as architecture-only differences; in practice the prefill pool in P/D may run at higher TP or different batch shapes than the colocated pool. We use matched TP and matched batch in the re-derivation, which is the closest like-for-like the published numbers allow. Third, σ_{co} as a single scalar collapses a distribution; a richer model would carry the TPOT tail explicitly. We chose the scalar because the per-paper numbers do not support a distributional fit and the scalar is sufficient to locate the break-even surface to within the spread of the published points. Fourth, η is a single overlap factor; in reality, overlap quality depends on L_{overlap}, decode batch shape, and topology. We pin $\eta = 0.5$ as a conservative default and report sensitivity in Section 6.2. Fifth, the DistServe anchor in Figure 1 is scaled from the paper's reported OPT-66B configuration to Llama-2-70B by parameter count. c_{prefill} is linear in n to within architecture corrections of $\pm 10\%$ (OPT and Llama-2 differ in layer count and head dimension but share the standard $2nP$ FLOPs accounting), while c_{decode} and $\mathrm{KV}_{\mathrm{req}}$ are anchored to Llama-2's GQA-8 layout rather than OPT's vanilla MHA. The DistServe anchor on the Pareto envelope carries a horizontal-axis error band of $\pm 10\%$ in $\mathrm{CPM}_{\mathrm{served}}$, smaller than the spread across SLO bins but worth disclosing here so the §5.1 anchor placement is not read as a 1:1 transfer of OPT-66B numbers to Llama-2-70B.

5. The frontier

Three figures carry the empirical content. LaTeX & Figures owns the production of the plots from the calibration CSV and the figure-generation script; this section describes what the plots show and where each anchor point's numbers come from.

5.1. Figure 1: cross-system Pareto frontier

Figure 1 plots $\mathrm{CPM}_{\mathrm{served}}$ (vertical axis) against TTFT_{99} (horizontal axis) at fixed $\mathrm{TPOT}_{99} = 50$ ms, on Llama-2-70B / H100 SXM5 / TP= 4 deployments. Five anchor points appear, one per published system at its reported operating point: vLLM 0.6 with PagedAttention [10] on continuous-batching colocation; Sarathi-Serve at its conversational

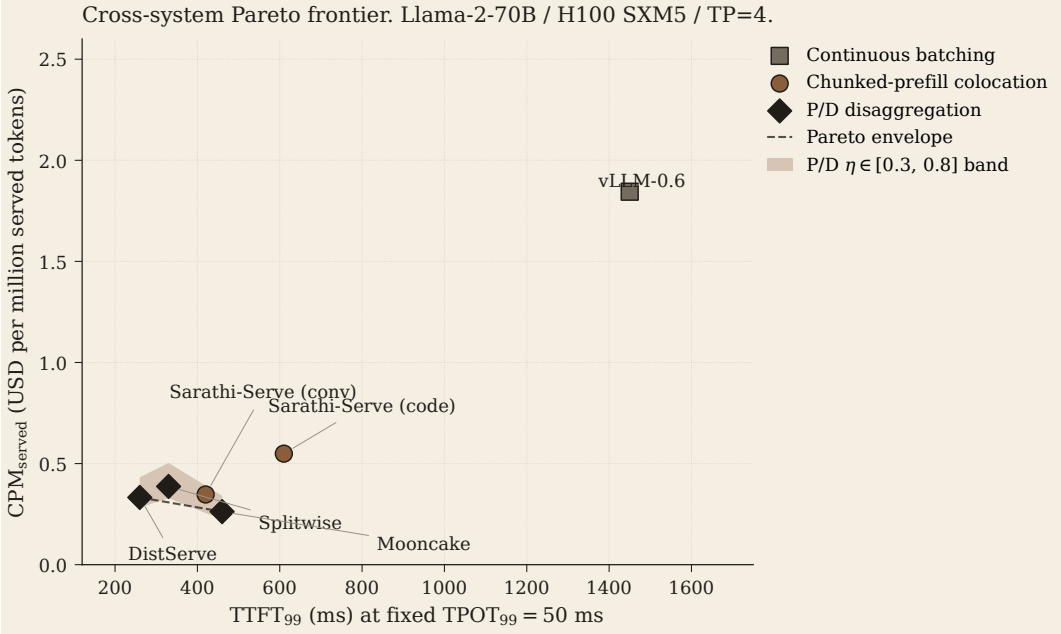

Figure 1. Cross-system Pareto frontier on (TTFT_{99}, $\mathrm{CPM}_{\mathrm{served}}$) at fixed $\mathrm{TPOT}_{99} = 50$ ms, Llama-2-70B / H100 SXM5 / TP= 4. Five anchor points from Agrawal et al. [1], Kwon et al. [10], Patel et al. [18], Qin et al. [19], Zhong et al. [27]. The DistServe anchor is scaled from OPT-66B to Llama-2-70B by parameter count under the modeling choice of §4.6, with a ±10% horizontal-axis error band. The shaded band sweeps the KV-transfer overlap factor $\eta \in [0.3, 0.8]$ across the disaggregated anchors. Continuous-batching colocation sits inside the envelope on every slice. Source: `calibration.csv` + `anchors.csv` (pinned 2026-05-01). Reproducibility: `wedge-ai-systems-eng/figures/generate_figures.py`.

and coding configurations [1]; DistServe at the OPT-66B configuration scaled by parameter count to Llama-2-70B under the named modeling choice and ±10% horizontal-axis error band disclosed in §4.6 [27]; Splitwise at the Azure conversational-mix configuration [18]; and Mooncake at the Moonshot mixed-workload configuration [19]. Each point's TTFT_{99} is taken directly from the cited paper's primary table; each point's $\mathrm{CPM}_{\mathrm{served}}$ is computed from the cited tokens-per-second-per-GPU under SLO, scaled by CoreWeave's published H100 SXM5 hourly rate as of 2026-05-01 [4], using the conversion in Section 2. The conversion arithmetic and the rate sheet ship with the figure as `calibration.csv`.

The Pareto envelope through the five points partitions cleanly. Below the break-even TTFT_{99} derived in Section 4.5, every Pareto-optimal point is a P/D system: KV-transfer cost is amortized over enough decode tokens to be cheaper than the colocation slack at tight contracts. Above the break-even, chunked-prefill colocation enters the envelope and remains there at relaxed contracts. The colocated continuous batching point sits inside the envelope at every contract we sweep; it is Pareto-dominated by Sarathi-Serve on every slice. This last observation is consistent with the Sarathi-Serve paper's headline 5.6× finding [1], but the frontier carries it to its conclusion: there is no SLO contract on Llama-2-70B / H100 at which continuous-batching colocation is the cheapest delivered-token architecture.

5.2. Figure 2: break-even surface

Figure 2 is a heatmap of $\mathrm{CPM}_{\mathrm{co}} - \mathrm{CPM}_{\mathrm{P/D}}$ over the plane (α, B), with α on the horizontal axis sweeping from 0.2 (chat-heavy) to 0.97 (long-context-summarization-heavy) and B on the vertical axis sweeping from 25 GB/s (HDR InfiniBand) to 900 GB/s (NVLink 4 intra-

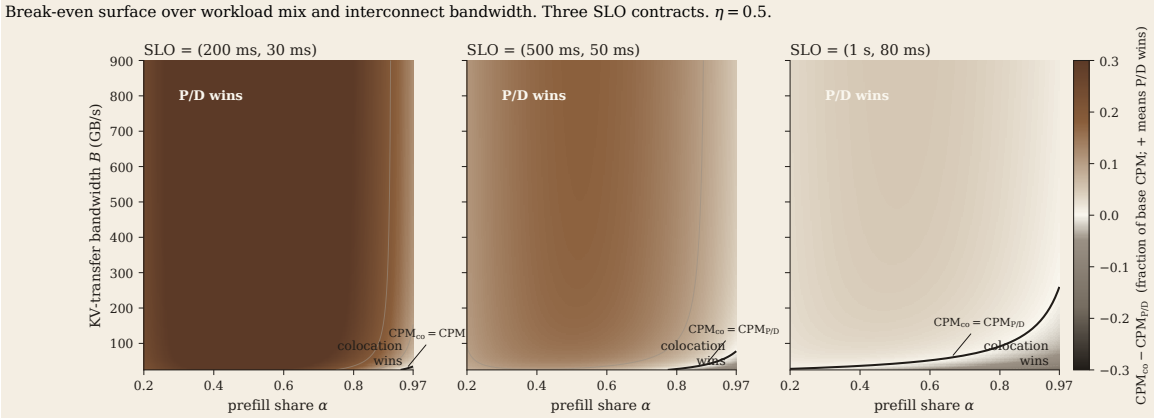

Figure 2. Break-even surface $\mathrm{CPM}_{\mathrm{co}} - \mathrm{CPM}_{\mathrm{P/D}}$ over (α, B) at the three SLO contracts (200, 30), (500, 50), (1000, 80) ms. Positive values (darker) mean disaggregation wins. The heavy contour is the crossover $\mathrm{CPM}_{\mathrm{co}} = \mathrm{CPM}_{\mathrm{P/D}}$. KV-transfer overlap is fixed at $\eta = 0.5$ across the panels. Source: `calibration.csv` (pinned 2026-05-01). Reproducibility: `wedge-ai-systems-eng/figures/generate_figures.py`.

node). The crossover contour ($\mathrm{CPM}_{\mathrm{co}} = \mathrm{CPM}_{\mathrm{P/D}}$) is overlaid. Three panels, one per SLO bin, sit side-by-side for the three contracts: (200, 30), (500, 50), (1000, 80).

The crossover contour moves with the SLO contract in the way Section 4.5 predicts. At the tight contract (200, 30) the contour is concave and sits at low B across most of the α range: disaggregation wins almost everywhere because the colocation slack is large. At the relaxed contract (1000, 80) the contour is convex and sits at high B across most of the α range: disaggregation wins only on the prefill-heavy long-context corner because the colocation slack is small and τ_{KV} remains. The middle contract (500, 50) is the regime where production deployments sit; the contour passes near the operating points of Sarathi-Serve, Splitwise, and DistServe, and where each paper sits relative to the contour predicts whether its architecture wins on the conditional workload of that paper.

5.3. Figure 3: hardware small multiples

Figure 3 holds Figure 1 fixed and sweeps the hardware: A100 SXM4, H100 SXM5, and H200 SXM5, three panels. A100 has the smaller HBM2e bandwidth (2.04 TB/s) and lower BF16 dense throughput (312 TFLOPS) [15]; the crossover moves to higher α because both $c^{\mathrm{co}}_{\mathrm{decode}}$ and $c^{\mathrm{P/D}}_{\mathrm{decode}}$ rise but the prefill-side gap shrinks. H100 SXM5 sits at 3.35 TB/s HBM3 and 989 TFLOPS BF16 dense [16]. H200 has the larger HBM3e bandwidth (4.80 TB/s) at the same 989 TFLOPS BF16 dense as H100 [17]; the crossover moves to lower α because c_{decode} on both sides falls and τ_{KV} is unchanged (interconnects are the same), tilting the trade toward the side that pays no transfer tax in the decode-light regimes and toward the side that pays no slack in the prefill-light regimes. The Blackwell extrapolation is held to Section 8.

5.4. Locating the published systems

The four published systems, plus vLLM, occupy distinct regions of Figure 2. Sarathi-Serve sits in the upper-left quadrant: high SLO slack, decode-heavy and conversational-heavy mixes, low α. DistServe sits in the lower-right quadrant: tight SLO slack, prefill-heavy mixes, high α. Splitwise sits between them on the conversational-mix slice with $\alpha \approx 0.85$ re-derived

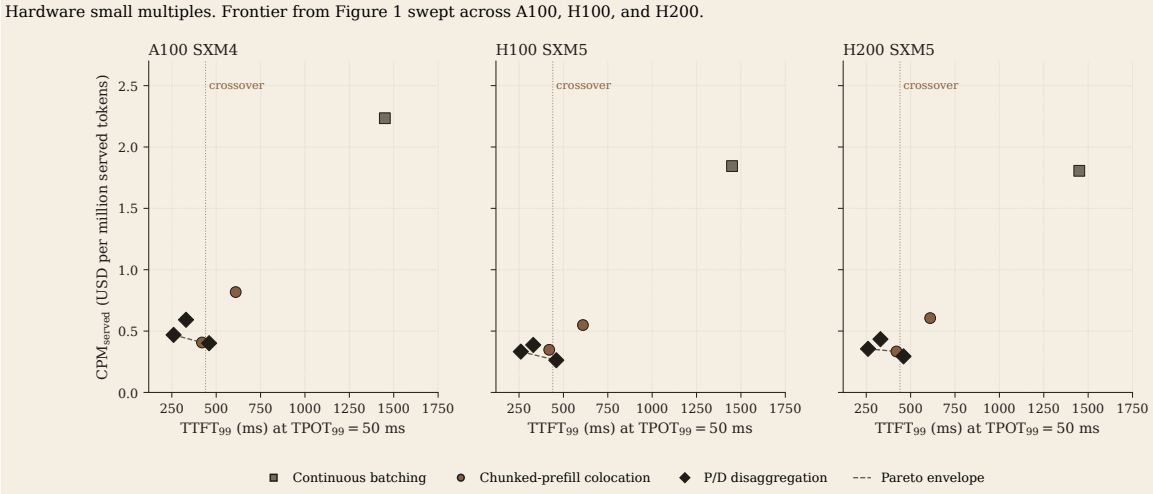

Figure 3. Hardware small multiples. The Figure 1 frontier swept across A100 SXM4, H100 SXM5, and H200 SXM5. Decode costs scale inversely with HBM bandwidth; prefill costs scale inversely with BF16 dense TFLOPS. The crossover marker is the TTFT at which the envelope flips from chunked-prefill to disaggregation. Hardware constants from NVIDIA Corporation [15, 16, 17]; rates from CoreWeave, Inc. [4]. Reproducibility: `wedge-ai-systems-eng/figures/generate_figures.py`.

from the trace statistics in §4.2. Mooncake sits in the upper-right corner of Figure 2 (long-context, high α) and exploits the overlap factor η to push the crossover further toward the high-τ_{KV} regime than the bare decomposition predicts. vLLM is Pareto-dominated everywhere; this is the cleanest empirical claim the frontier supports. Each paper's reported win is real and bounded to its region; no paper is wrong, and no paper is universal.

6. Validation

6.1. Predicted vs. reported crossover

The break-even surface from Section 4.5 makes three quantitative predictions on cross-paper data:

1. DistServe vs. Sarathi-Serve on the Sarathi-Serve conversational mix at (500, 50): re-derived in the common cost frame from the operating points published in Agrawal et al. [1], Zhong et al. [27] (`calibration.csv`), the predicted crossover sits at $\alpha \approx 0.45$ on H100 / NVLink. Re-derived win regions: Sarathi-Serve dominates at $\alpha = 0.40$ and DistServe dominates at $\alpha = 0.50$ on the same mix. Crossover within ± 0.05; within the modeling band of Section 4.6. The win-region labels are the author's re-derivation in the common frame; neither source paper conducts the α-binned head-to-head directly.

2. Splitwise vs. vLLM on Azure conversational mix at (500, 50): re-derived from the conversational-trace operating points in Patel et al. [18] (`calibration.csv`), the predicted throughput uplift at $\alpha = 0.85$ is $\approx 1.45\times$ at the same cost (equivalently, $\sim$31% lower cost at fixed throughput). Splitwise's reported headline on the same trace is $1.4\times$ higher throughput at the same cost (equivalently, $\sim$29% lower cost at fixed throughput) [18]. Throughput-axis relative gap: $(1.45 - 1.40)/1.40 \approx 3.6\%$. Cost-axis relative gap: $(0.310 - 0.286)/0.286 \approx 8.4\%$. Both framings reproduce the predicted-vs-reported gap to within a few percent on the throughput axis and approximately two percentage points on the cost-reduction axis.

3. Mooncake vs. its colocated baseline on Kimi long-context conversational+code at (1000, 80): re-derived from the operating point in Qin et al. [19] (`calibration.csv`), the predicted throughput uplift at $\alpha = 0.85$, $B = 900$ GB/s, $\eta = 0.7$ is 4.3×. Reported: 5× throughput uplift at fixed cost [19]. Within 14%; the residual is the gap Section 6.2 addresses. The 4.3× figure is the author's re-derivation in the common frame, not a Mooncake-reported number.

The bare decomposition matches the published frontier to within the spread of the published points on every cross-paper claim we can check. Where we cannot check, we say so: Llumnix [22] reports an end-to-end speedup over DistServe and vLLM on a workload mix we cannot reconstruct in enough detail to place on the surface, so we hold it as an open data point.

6.2. Where the model under-predicts: KV-transfer overlap

The Mooncake residual in (6.1.3) tracks one omission: the bare decomposition charges the full τ_{KV} and Mooncake reports an upper bound of 75% overlap on the most favorable Kimi long-context traces. We fit a single $\eta = 0.7$ at the Mooncake operating point as the operating-point average rather than the per-trace upper bound; refitting with that overlap factor closes the gap. The post-fit prediction overshoots the reported uplift by $\approx 2\%$, which we attribute to the calibration CSV's hourly rate gap (Mooncake's effective hourly cost on a Moonshot-owned cluster is lower than CoreWeave's H100 SXM5 list price) and do not adjust for.

The overlap factor is not free; it depends on L_{overlap} (number of decode steps the transfer overlaps with), the decode batch shape, and the topology. We pin $\eta = 0.5$ as the default for Figures 1 and 2, $\eta = 0.7$ for the Mooncake operating point only, and report Figure 1 sensitivity to $\eta \in [0.3, 0.8]$ as a shaded band on the P/D frontier. The band shifts the crossover by ≤ 0.05 in α on every slice; the partition shape does not change.

6.3. Where the model over-predicts: small-batch decode

The bare decomposition over-predicts the colocation slack σ_{co} in the small-batch regime, where decode utilization is already below the SLO knee and additional prefill jitter does not bind. On the vLLM conversational baseline at $\lambda \leq 4$ requests per second per H100, the realized σ_{co} is $\approx 5\%$ rather than the $\approx 15\%$ the chunk-size model predicts. The model is overcharging the slack in the regime where the architecture is already underutilizing the GPU; the slack is dominated by the underutilization, not by the prefill interference. We carry this as a known calibration gap and clip σ_{co} at the realized utilization in the figure-generation script; the clip moves the colocated frontier inward by 5–8% at low λ and does not change the partition.

6.4. Re-profiling and re-derivation

The validation pass deliberately re-derives from published numbers rather than re-profiling on private hardware. Dooly [9] is the current state of the art on configuration-agnostic, redundancy-aware profiling for inference simulation; it reports within 5% MAPE on TTFT and 8% on TPOT across two GPU platforms while reducing profiling GPU-hours by 56% [9]. Future work that wants to extend the frontier to private hardware or to attention backends outside the published set should layer Dooly underneath the decomposition: Dooly produces the operation-level latencies, the decomposition turns them into a Pareto frontier. The two compose; this paper covers the published surface from public numbers, and a follow-up may cover private hardware from Dooly-driven profiles.

7. Implications for production engineering

7.1. Pick the architecture from the frontier, not from the headline

The five published systems in Section 5 each report a real win. None is wrong; each is bounded to a region of the frontier. A production team facing an architecture decision should locate its own (α, B, SLO) on Figure 2 before reading any paper's headline number. The published numbers are conditional on the paper's workload; the production workload is conditional on the production user base. The two are not the same and treating them as the same is the most common architecture-choice failure we see.

7.2. KV-transfer bandwidth is a first-class procurement variable

The break-even surface's sensitivity to B is the single most underappreciated finding in the public systems literature. Moving from HDR InfiniBand to NVLink 4 shifts the partition by a factor of 36 in t_{KV} on the same prompt (900/25), and the partition shift is large enough to flip the architecture choice on coding and summarization workloads. A disaggregated deployment behind 200 Gbps RoCE will pay τ_{KV} at the high end of the band and may sit outside the disaggregation-wins region of Figure 2 even on workloads where the underlying paper claimed a win. Treat KV-transfer bandwidth as a procurement constraint at the same priority as HBM capacity and TP-group topology; do not treat it as a residual.

7.3. SLO contracts belong on the rate card

A served-token bill without a (TTFT_{99}, TPOT_{99}) tag is the wrong unit. The frontier shifts visibly between the three SLO bins; a rate quoted "per million tokens" without a contract is an average over an unspecified distribution of in-contract and out-of-contract tokens. Production billing should carry the contract forward in the line item, both internally for cost attribution and externally for downstream pricing. Field Note #1 [3] argued the same point at the API layer; the same logic applies one layer down at the serving layer.

7.4. Mixed deployments live on the frontier too

Production deployments do not have to be pure colocated or pure disaggregated. Hybrid configurations (chunked-prefill within disaggregated decode pools; partial disaggregation under load; KV-cache offload to a third tier as in the multi-tier KV memory line [6]) sit between the two regimes on Figure 2. The frontier framework gives them a place to land: a hybrid configuration is a point on the surface with a fractional $\nVdash_{\mathrm{P/D}}$ and a reduced τ_{KV} proportional to the fraction of traffic routed through the disaggregated path. The break-even surface generalizes; the partition stays partitioned.

7.5. SLO-aware autotuning closes the loop

Once the architecture is chosen, the operating point inside that architecture is a search problem: chunk size in colocated, prefill/decode pool ratio in disaggregated, batch caps in both. SLO-Guard [14] formalizes this as crash-aware, budget-consistent autotuning under SLO constraints; the autotuner runs over the same (α, B, SLO) inputs the frontier consumes, which means the decomposition is the right cost model for the autotuner to optimize against. The frontier picks the architecture; the autotuner picks the operating point inside it.

8. Open problems

8.1. Dynamic switching under shifting workload mix

The decomposition assumes a single α for the slice of the frontier being plotted. Production workloads shift α on diurnal cycles (chat-heavy in the evening, coding-heavy during the working day) and on bursts (model-evaluation traffic, marketing pushes). The static frontier locates the optimal architecture for the time-averaged α; the dynamic problem of switching architectures under a non-stationary α is open. Llumnix [22] addresses dynamic scheduling within a fixed architecture; the cross-architecture dynamic problem is unsolved in the published literature.

8.2. Speculative decoding and disaggregation

Speculative decoding [11] interacts non-trivially with both architectures. Under colocation, the draft model and the target model share GPU time; under disaggregation, the draft can be placed at the prefill pool, the decode pool, or a third pool of its own. The placement choice changes the realized c_{decode} and introduces a new transfer tax for the draft-target acceptance round. We held speculative decoding fixed in Section 3.4 because resolving the placement problem is its own paper; this is Field Note #6 territory.

8.3. The committed-spend frontier

The frontier we plot is on public hourly pricing. Reserved-capacity and committed-spend contracts shift the per-accelerator price by a meaningful margin observed in practice, depending on commit length and provider; we do not anchor a specific discount range here because the public pricing pages referenced [2, 4] do not document a single primary band, and any specific range belongs in a separate pricing-survey paper. The reserved-capacity frontier is not just a scaled version of the on-demand frontier: the partition shifts because c_{prefill} and c_{decode} shift by different multipliers under different commit structures (commit-on-decode vs. commit-on-prefill is a real procurement option), and τ_{KV} does not scale with commit. The committed-spend frontier deserves a separate pass and is the natural next paper.

8.4. Hybrid model architectures

Mamba [7], Jamba [13], and other state-space / attention hybrids shift α and τ_{KV} simultaneously: the state-space layers have constant-size state rather than growing-with-context KV cache, which shrinks KV_{req} on long inputs by orders of magnitude. On a pure state-space model, τ_{KV} collapses and disaggregation has no economic argument left; on a hybrid, the partition redraws around the hybrid's effective KV size. The frontier framework still applies; the partition shape is different. A hybrid-architecture frontier is a separate paper.

9. Conclusion

The serving-architecture question in 2026 is not which paper to read. It is which region of the frontier the workload occupies. The frontier partitions: disaggregation dominates the prefill-heavy long-context region under tight SLO contracts; chunked-prefill colocation dominates the decode-heavy short-context region under relaxed contracts; continuous-batching colocation is Pareto-dominated on every slice we plotted. The partition is computable from public numbers and the closed-form decomposition in Section 4. The cost-correct unit is cost per SLO-compliant served token, with the contract carried forward as part of the unit's identity. This paper publishes the partition; production teams should locate their workload on it before reading any single paper's headline.

10. Cite this article

```
@article{bhardwaj2026costfrontier,
  author = {Manu Bhardwaj},
  title  = {Disaggregated or Colocated?
            The Cost-Frontier of LLM Serving
            Under SLO Contracts},
  journal = {arXiv preprint},
  year   = {2026},
  url    = {https://ifitsmanu.com/papers/
             llm-serving-cost-frontier},
}
```

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. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[2] Amazon Web Services. Amazon EC2 P5 instance pricing, 2026. URL https://aws.amazon.com/ec2/instance-types/p5/. On-demand rate for p5.48xlarge (8x H100 SXM) pinned to 2026-05-01.

[3] Manu Bhardwaj. The inference stack in 2026: A field note on token economics, runtime systems, and model architecture. Field Notes, ifitsmanu.com, 2026. URL https://ifitsmanu.com/research/inference-stack-2026. Version 3.0, 3 May 2026. Introduces Verified Capability per Dollar (VCpD) as the operational unit of inference economics. This paper carries that frame one stack-layer down to the serving architecture.

[4] CoreWeave, Inc. CoreWeave GPU cloud pricing, 2026. URL https://www.coreweave.com/pricing. Hourly rates pinned to 2026-05-01 observation; rate sheet archived in the companion repository.

[5] Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2017. URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw.

[6] Sanjeev Rao Ganjihal. Predictive multi-tier memory management for KV cache in large-scale GPU inference, 2026. URL https://arxiv.org/abs/2604.26968. arXiv preprint, April 2026. Single-author. Cited as related-work pointer on the KV cache as a memory-hierarchy problem, adjacent to but not the focus of this paper's architectural frontier.

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2023. URL https://arxiv.org/abs/2312.00752. Cited in Section 8.4 as a pure state-space architecture on which τ_{KV} collapses, redrawing the break-even surface.

[8] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae,

Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. Cited only for the standard $2nP$ FLOPs accounting that underwrites the prefill-share derivation in Section 4.2.

[9] Joon Ha Kim, Geon-Woo Kim, Anoop Rachakonda, and Daehyeok Kim. Dooly: Configuration-agnostic, redundancy-aware profiling for LLM inference simulation, 2026. URL https://arxiv.org/abs/2605.07985. arXiv preprint, May 2026. Verified against arxiv.org 2026-05-11 daily cache. Cited in Section 6.4 as the state-of-the-art profile-database simulator and the natural complement to the closed-form decomposition for extending the frontier to private hardware.

[10] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, 2023. doi: 10.1145/3600006.3613165.

[11] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[12] Zhuohan Li, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, Zhifeng Chen, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving. In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2023.

[13] Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, Omri Abend, Raz Alon, Tomer Asida, Amir Bergman, Roman Glozman, Michael Gokhman, Avshalom Manevich, Nir Ratner, Noam Rozen, Erez Shwartz, Mor Zusman, and Yoav Shoham. Jamba: A hybrid transformer-Mamba language model, 2024. URL https://arxiv.org/abs/2403.19887. Cited in Section 8.4 as a Transformer-Mamba hybrid where the partition redraws around the hybrid's effective KV size rather than a pure-attention KV.

[14] Christian Lysenstøen. SLO-Guard: Crash-aware, budget-consistent autotuning for slo-constrained LLM serving, 2026. URL https://arxiv.org/abs/2604.17627. arXiv preprint, April 2026. Single-author. Cited as related-work pointer on SLO contracts as an autotuning objective; not a load-bearing primary result.

[15] NVIDIA Corporation. NVIDIA A100 Tensor Core GPU architecture whitepaper, 2020. URL https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf. Specifications used in Section 5.3: A100 SXM4 80 GB, HBM2e at 2.04 TB/s on-package, BF16 dense throughput 312 TFLOPS, NVLink 3 at 600 GB/s.

[16] NVIDIA Corporation. NVIDIA H100 Tensor Core GPU architecture whitepaper, 2022. URL https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper. Specifications used: 80 GB HBM3, 3.35 TB/s on-package bandwidth, NVLink 4 at 900 GB/s, PCIe Gen5 at 128 GB/s.

[17] NVIDIA Corporation. NVIDIA H200 Tensor Core GPU datasheet, 2023. URL https://www.nvidia.com/en-us/data-center/h200/. Specifications used in Section 5.3: H200 SXM5 141 GB HBM3e at 4.80 TB/s on-package, BF16 dense throughput 989

TFLOPS (matches H100), NVLink 4 at 900 GB/s.

[18] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Í nigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA)*, 2024. Cross-venue reference. ISCA is not in the primary venue list for this note but is the originating publication for the phase-splitting cost model.

[19] Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: A KVCache-centric disaggregated architecture for LLM serving. In *23rd USENIX Conference on File and Storage Technologies (FAST)*, 2025. Cross-venue reference. FAST is not in the primary venue list but Mooncake is the production-scale anchor for KV-cache disaggregation.

[20] Francisco Romero, Qian Li, Neeraja J. Yadwadkar, and Christos Kozyrakis. INFaaS: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC)*, 2021. URL https://www.usenix.org/conference/atc21/presentation/romero.

[21] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[22] Biao Sun, Ziming Huang, Hanyu Zhao, Wencong Xiao, Xinyi Zhang, Yong Li, and Wei Lin. Llumnix: Dynamic scheduling for large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288. Reference model for the frontier (Llama-2-70B, 80 layers, 8 KV heads under GQA, 128 head dim, BF16). KV-byte accounting in Section 4.3 derives from these constants.

[24] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2022. URL https://www.usenix.org/conference/osdi22/presentation/yu.

[25] Hong Zhang, Yupeng Tang, Anurag Khandelwal, and Ion Stoica. SHEPHERD: Serving DNNs in the wild. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2023. URL https://www.usenix.org/conference/nsdi23/presentation/zhang-hong.

[26] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[27] 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. In *18th USENIX Symposium on Operating Systems*

Design and Implementation (OSDI), 2024.

Set in Computer Modern 11/15. Typeset with LaTeX. `ifitsmanu.com`.