ANALYTICAL PROVISIONING FOR ATTENTION–FFN DISAGGREGATED LLM SERVING UNDER STOCHASTIC WORKLOADS

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention–FFN disaggregation (AFD) is an emerging architecture for LLM decoding that separates state-heavy, KV-cache–dominated Attention computation from stateless, compute-intensive FFN computation, connected by per-step communication. While AFD enables independent scaling of memory and compute resources, its performance is highly sensitive to the Attention/FFN provisioning ratio: missizing induces step-level blocking and costly device idle time. We develop an analytical provisioning framework for AFD bundles in an rA–1F topology under stochastic workloads. Two sources of randomness shape the problem: per-slot Attention workload evolves as KV caches grow and completed requests are replenished with random prompt and decode lengths, and synchronized execution across Attention workers introduces a barrier governed by the slowest worker. We address both via a renewal-reward characterization of the per-slot stationary token load, identifying a single workload statistic θ that governs provisioning under arbitrary prefill-decode distributions and admits a nonparametric estimator from request traces. The analysis yields a closed-form mean-field rule for the optimal A/F ratio decomposing into Attention-, communication-, and FFN-bottleneck regimes, together with a Gaussian barrier-aware refinement that quantifies cross-worker synchronization overhead. A trace-calibrated AFD simulator supports the framework across workloads: the predicted optimal ratio matches the simulation-optimal within 10%. Together, these results provide a compact, calibratable account of how stochastic workload structure determines provisioning in disaggregated LLM serving.

1 INTRODUCTION

The explosive growth of large language models (LLMs) (Brown et al., 2020; Chowdhery et al., 2023; OpenAI, 2023; Kaplan et al., 2020) has created unprecedented challenges for efficient inference serving at scale. As model sizes expand into hundreds of billions of parameters, modern LLM serving inevitably requires distributed multi-device architectures. This reality has spurred a shift from monolithic serving paradigms toward increasingly sophisticated disaggregation strategies that decompose the inference pipeline across heterogeneous hardware resources.

Early efforts of disaggregate serving focused on stage-level separation, exemplified by the Prefill-Decode (PD) disaggregation (Zhong et al., 2024; Patel et al., 2024). LLM inference comprises two phases with distinct resource profiles: *prefill* processes the input prompt and generates the KV cache via dense matrix operations (compute-bound), while *decode* generates tokens autoregressively, reading the accumulated KV cache at each step (memory-bound). PD disaggregation separates these phases onto hardware matched to their respective bottlenecks. By disaggregating prefill and decode onto separate hardware pools that can be independently scaled, PD architectures enable higher decode batch sizes through request aggregation, improving GPU utilization during token generation.

Building upon this foundation, recent research has recognized that even within the decode phase, computational heterogeneity persists between different transformer components. The Attention-FFN Decoupling (AFD) (Wang et al., 2025; Zhang et al., 2025; Zhu et al., 2025; Zuo et al., 2025) extends this disaggregation further: Attention layers are stateful and memory-bound (dominated by KV cache

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reads), whereas FFN layers are stateless and can achieve compute-bound operation with sufficient batching. Traditional coupled architectures often leave FFN compute units underutilized due to the small decode batch with TPOT constraints. AFD architecture strategically disaggregates these components, allowing multiple Attention instances to feed into an aggregated FFN instances. This topology maximizes FFN arithmetic intensity while independently scaling memory resources for Attention.

Recent work (Wang et al., 2025; Zhu et al., 2025; Zuo et al., 2025) has shown that AFD can significantly improve serving efficiency by allowing Attention and FFN to scale independently. Given this flexibility, a central design decision in such systems is the ratio of Attention instances to FFN instances, which we denote as r. Currently, existing systems set r through empirical search or naive deterministic approximations that ignore workload stochasticity. A principled analytical provisioning framework is still lacking. This gap matters because the choice of r directly impacts system efficiency: when r is too small, the FFN server starves for input; when r is too large, Attention instances block waiting for FFN availability.

Finding a principled analytical provisioning rule, however, is non-trivial because the Attention workload is fundamentally stochastic. FFN computation depends only on batch size and remains stable across decode steps. In contrast, Attention workload evolves continuously—the KV cache grows with each step, and completed requests are replaced by new ones with variable prompt lengths.

As AFD remains an emerging paradigm with no mature open-source implementations, principled **analytical foundations** are needed to guide this design space before extensive system building begins. In this paper, we provide such foundations through a rigorous probabilistic framework that captures the stochastic dynamics of Attention-side workload and derives a closed-form expression for the optimal A/F ratio. Our simulation-based validation demonstrates that the theory accurately predicts system behavior; real-system validation is left to future work as AFD implementations mature. Our contributions are summarized as follows:

(i) Probabilistic workload model. We develop an analytical framework capturing the stochastic dynamics of AFD serving under continuous batching with request replenishment. Using a discrete-time renewal-reward analysis of one decode slot, we identify a single workload statistic that governs provisioning under arbitrary prompt and decode-length distributions and admits a nonparametric estimator from request traces.

(ii) Provisioning rule with load-balancing correction. We derive a closed-form mean-field rule for the A/F ratio that decomposes into three operating regimes (Attention-bottleneck, communication-bottleneck, FFN-bottleneck), each with an interpretable balance condition. Cross-worker synchronization stragglers are handled via a Gaussian order-statistic correction.

(iii) Validation across workloads and hardware regimes. We develop a trace-calibrated AFD simulator and validate the framework across diverse workload and hardware configurations. The closed-form rule matches the simulator-optimal ratio within 10%, which is an empirical confirmation of the theory.

2 Background on Attention-FFN Disaggregation: Pipeline and Microbatch

In standard LLM decoding, a monolithic architecture deploys both Attention and FFN blocks on the same hardware. To reduce kernel launch overhead and execution bubbles, practitioners typically compile the model into a static execution graph (e.g., CUDA Graphs (Ekelund et al., 2025)), which enforces fixed batch sizes across all components. This rigidity creates a utilization problem: time per output token (TPOT) latency constraints force small batch sizes, and at low concurrency, FFN blocks cannot amortize their weight-loading cost—despite being computationally dense, they become memory-bandwidth bound rather than compute-bound.

AFD addresses this by disaggregating Attention and FFN onto separate hardware, enabling an asymmetric $r : 1$ deployment where multiple Attention instances feed a shared FFN server. We now describe the execution mechanics of this architecture.

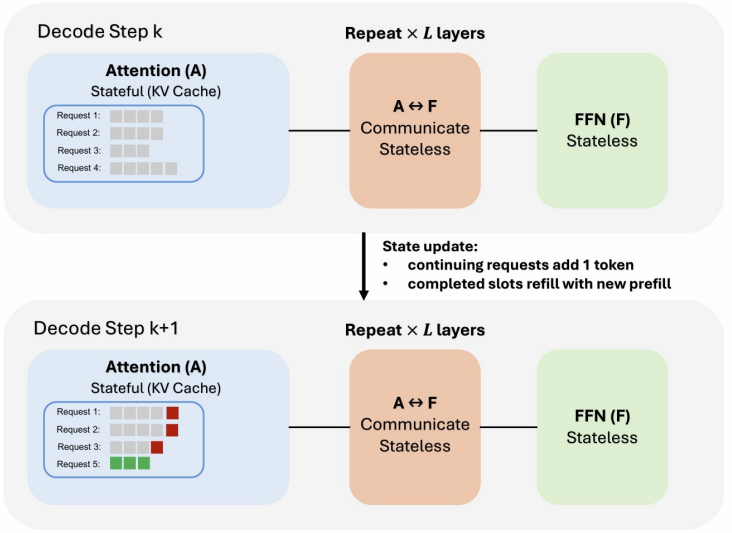
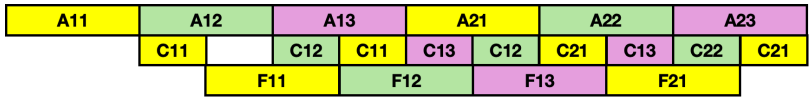
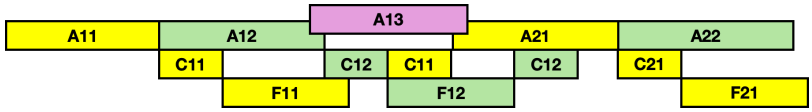


Figure 1: Architecture of AFD. Stateful Attention (A) layers manage the KV cache, while Feed-Forward Network (F) layers are stateless. During each decode step, every continuing request generates one output token whose key-value is appended to the KV cache (red blocks); when a request completes, its slot is immediately refilled with a new prefill request (green block).



(a) Ideal microbatch



(b) Issue raised after one decode step

Figure 2: Microbatch Pipelining and Masking. **(a)** An ideal schedule where Attention, Communication, and FFN are perfectly overlapped across microbatches, fully hiding data transfer latency. **(b)** After one decode step, Attention execution time increases (due to longer KV cache reads) while Communication and FFN remain unchanged, breaking the balanced overlap and introducing pipeline bubbles.

In an AF-disaggregated deployment, each Transformer layer comprises three distinct stages: Attention computation, FFN computation, and inter-device Communication ($A \leftrightarrow F$). As illustrated in Figure 1, Attention blocks are stateful: during each decode step, every continuing request generates one output token, and its corresponding key-value is appended to the KV cache. When computing attention, the model must read the entire accumulated KV cache for each request, so the Attention cost grows linearly with sequence length. When a request completes, its slot is immediately refilled with a newly prefill request to maintain batch size (continuous batching). In contrast, FFN blocks are stateless—their computation depends only on the current activations, not on sequence history. To handle the latency introduced by transferring activations (AF communication) across L layers within one decode step, microbatch pipelining is adopted, as shown in Figure 2.

Figure 2(a) depicts an ideal execution schedule. Here, A_{mn}, C_{mn}, and F_{mn} denote Attention, Communication, and FFN operations for the n-th microbatch at the m-th layer, respectively. With sufficient microbatches (typically ≥ 3), the system overlaps Communication and FFN of one microbatch with Attention of subsequent microbatches, thereby masking data transfer (communication) latency.

However, Attention's stateful nature introduces dynamic variability that disrupts this ideal schedule. Consider Figure 2(b): after one decode step, assuming no request completes, every request's KV cache has grown by one token. Consequently, A_{11}, A_{12}, and A_{13} all take longer than in the previous step, while C_{mn} and F_{mn} remain unchanged. The increased Attention time means that A_{12} and A_{13} can no longer fit in the gaps between layers—A_{13} is deferred, leaving idle "bubbles" in the pipeline. Depending on the workload imbalance, bubbles may appear on the Attention side (waiting for FFN) or the FFN side (starved of data from Attention). Since Attention workload drifts continuously, no static microbatch schedule remains ideal over time. This motivates our central question: how to determine the optimal Attention-to-FFN ratio (r:1) that minimizes expected pipeline bubbles across a long serving horizon.

3 Mathematical Model

We consider an AFD bundle in a general xA–yF topology, where x Attention instances feed into y shared FFN instances. To simplify notation, we define the Attention-to-FFN ratio $r := x/y$ and express the topology equivalently as rA–1F. r does not need to be an integer, for example, $r = 3.5$ corresponds to a 7A–2F configuration. Each Attention instance maintains a microbatch of B requests and their associated KV caches. Decoding proceeds in synchronized steps, where each step comprises four phases:

(i) *Attention phase:* The r Attention workers execute in parallel, each processing its microbatch of B requests (total rB requests across the bundle).

(ii) *A→F communication:* All r Attention workers transfer their intermediate activations to the shared FFN worker.

(iii) *FFN phase:* The FFN worker processes the aggregated batch of rB activations.

(iv) *F→A communication:* The FFN worker returns outputs to all r Attention workers.

Our goal is to determine the optimal ratio r^* that maximizes system throughput.

3.1 Latency Models

We adopt linear latency models grounded in first-principles hardware analysis (see Appendix B for detailed derivations) and consistent with established results in the LLM inference literature.

Attention. The Attention computation reads the KV cache for all requests in the microbatch. For a microbatch of B requests, let s_b denote the prefill length and i_b the current decode index for request b. The total token load is $T = \sum_{b=1}^{B}(s_b + i_b)$. During decoding, Attention is memory-bandwidth bound—dominated by reading the KV cache from HBM (Kwon et al., 2023; Pope et al., 2023; Agrawal et al., 2024). By the roofline model, latency for a memory-bound operation equals data volume divided by effective bandwidth (Yuan et al., 2024; Lienhart, 2024), so Attention latency scales linearly with T: $t_A(T) = \alpha_A T + \beta_A$.

FFN. The FFN processes batched activations and becomes compute-bound with sufficient batching (Agrawal et al., 2024; Wei et al., 2024). FFN layers perform matrix multiplications whose FLOPs scale as $O(\text{batch} \times \text{hidden} \times \text{intermediate})$, yielding linear latency in batch size (Pope et al., 2023; Chen, 2023): $t_F(rB) = \alpha_F(rB) + \beta_F$.

Communication. The round-trip communication cost per Attention instance is $t_C(rB) = \alpha_C(rB) + \beta_C$, where β_C is a fixed startup cost and $\alpha_C(rB)$ is the bandwidth-dependent term. This is the standard model for distributed communication, validated extensively in GPU clusters (Narayanan et al., 2021; Li et al., 2024).

3.2 Workload Characterization

Request lengths. Each request has a prefill length $P \in \mathbb{Z}_{\geq 0}$ and a decode lifetime $D \in \{1, 2, \ldots\}$, where D is the number of decode steps the request occupies its slot. During those D steps, the decode index takes values $0, 1, \ldots, D-1$. We treat $(P_n, D_n)_{n \geq 1}$ as i.i.d. across requests, allowing arbitrary dependence between P_n and D_n within a request. We assume only that $\mathbb{E}[P]$, $\mathbb{E}[D]$, and the second

moment of the per-slot stationary load (defined in Section 4.1) are finite. No specific distributional family is required.

Continuous batching. Following standard practice in LLM serving, we use continuous batching: whenever a request completes, its slot is immediately refilled with a new request from the queue, maintaining full microbatches throughout serving.

Cross-worker load balancing. The r Attention workers in a bundle execute synchronously, so the Attention phase waits for the slowest worker. Stochastic request dynamics induce heterogeneous token loads across workers, creating a straggler effect that grows with r. Implementing load-balancing routing policies (Chen et al., 2026) can mitigate this gap, though some irreducible variance from stochastic dynamics remains.

3.3 Optimization Objective

Cycle time. Let T_j denote the total token load on Attention worker $j \in \{1, \ldots, r\}$. Because the r workers execute synchronously, the Attention phase completes only when the slowest worker finishes. The relevant Attention-side load is therefore the *barrier load* $W_{B,r} := \max_{1 \le j \le r} T_j$. The per-step cycle time is $\tau(B; r) = \max\left\{\alpha_A W_{B,r} + \beta_A,\ t_C(rB),\ t_F(rB)\right\}$.

Throughput per instance. Long-run throughput is governed by the expected cycle time $\bar{\tau}(B; r) := \mathbb{E}[\tau(B; r)]$. We maximize the average number of output tokens generated per unit time, normalized by the total number of instances (r Attention + 1 FFN):

$$\text{Throughput}_{\text{per-inst}}(B; r) = \frac{1}{r+1} \cdot \frac{rB}{\bar{\tau}(B; r)}, \tag{1}$$

where rB is the number of tokens generated per cycle across the bundle.

The central challenge is that $\bar{\tau}(B; r)$ is shaped by two sources of stochasticity. Within each Attention worker, per-slot loads evolve as decoding progresses and slots are replenished with random prompt and decode lengths, so the workload is non-stationary at the step level even though it admits a stationary distribution under continuous batching. Across workers, the maximum in $W_{B,r}$ introduces a synchronization barrier whose expected overhead grows with r. The next section develops a probabilistic analysis to provide a provisioning rule.

4 Analysis and Optimal A/F Ratio

We now derive the optimal Attention-to-FFN ratio r^*. The analysis must contend with two sources of randomness simultaneously: per-slot Attention workload evolves pathwise as decoding progresses, with KV caches growing and completed slots replenished by new requests of random length, and the synchronized Attention phase is governed by a barrier across the r workers. Classical fork-join queueing models assume stationary, identical service times and therefore do not directly apply. We address both effects by combining a renewal-reward characterization of the per-slot stationary load with a Gaussian order-statistic correction for the cross-worker barrier. The analysis yields a closed-form mean-field provisioning rule, refined by a barrier-aware correction that captures synchronization overhead.

4.1 Stationary Per-Slot Token Load

By symmetry, it suffices to analyze one decode slot under continuous batching. Requests arrive at the slot sequentially. Request n has prompt length $P_n \in \mathbb{Z}_{\ge 0}$ and decode lifetime $D_n \in \{1, 2, \ldots\}$, with $(P_n, D_n)_{n \ge 1}$ i.i.d. across requests. During its lifetime, the request occupies the slot for D_n decode steps, contributing token load $P_n + a$ at age $a \in \{0, 1, \ldots, D_n - 1\}$. Let Y denote the stationary token load of one slot, observed at a uniformly random decode step, and define $\theta := \mathbb{E}[Y]$, $\nu^2 := \text{Var}(Y)$.

Lemma 4.1 (Stationary token-load moments)**.** *Assume* $0 < \mathbb{E}[D] < \infty$ *and*

$$\mathbb{E}\Big[DP^2 + PD(D-1) + \tfrac{D(D-1)(2D-1)}{6}\Big] < \infty. \tag{2}$$

Then

$$\theta = \frac{\mathbb{E}\Big[DP + \frac{D(D-1)}{2}\Big]}{\mathbb{E}[D]}, \qquad \mathbb{E}[Y^2] = \frac{\mathbb{E}\Big[DP^2 + PD(D-1) + \frac{D(D-1)(2D-1)}{6}\Big]}{\mathbb{E}[D]}, \tag{3}$$

and $\nu^2 = \mathbb{E}[Y^2] - \theta^2$. *If* P *and* D *are independent,*

$$\theta = \mu_P + \frac{\mu_D - 1}{2} + \frac{\sigma_D^2}{2\mu_D}, \tag{4}$$

where $\mu_D := \mathbb{E}[D]$ *and* $\sigma_D^2 := \mathrm{Var}(D)$. *Without independence, an additional term* $\mathrm{Cov}(P, D)/\mu_D$ *enters.*

The proof, given in Appendix A.1, applies the discrete-time renewal-reward theorem to one decode slot, treating each request as a renewal cycle of length D_n.

Equation (4) identifies the workload statistic that drives provisioning: it is *not* the arrival-average $\mu_P + \mu_D$ (a natural but incorrect first guess), but the stationary age-adjusted load. The decode-length variance σ_D^2 enters through length-biasing (longer requests are sampled more often when observed at a random decode step) and the prefill–decode covariance enters when long prompts induce long responses. The framework imposes no parametric assumption on (P, D); Appendix A.6 gives a nonparametric estimator that calibrates θ and ν^2 directly from request traces.

Remark 4.2 (Bounded request lengths). In practice, serving systems impose maximum context and generation lengths, so P and D are bounded and condition (2) is automatically satisfied. Appendix A.7 discusses the role of length-biasing under unbounded distributions for completeness.

4.2 Barrier-Aware Attention Load

Each Attention worker $j \in \{1, \ldots, r\}$ holds B slots with stationary loads $Y_{j,b}$, i.i.d. across (j, b). The total load on worker j is $T_j^{(B)} = \sum_{b=1}^{B} Y_{j,b}$, and the synchronization barrier load is $W_{B,r} = \max_{1 \le j \le r} T_j^{(B)}$. Throughout this subsection we treat r as a positive integer. Let $Z_1, \ldots, Z_r \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$ and $M_r := \max_j Z_j$, with

$$\kappa_r := \mathbb{E}[M_r] = \int_{-\infty}^{\infty} z\, r\phi(z)\Phi(z)^{r-1}\, dz, \tag{5}$$

where ϕ and Φ are the standard normal density and CDF. For large r, $\kappa_r \sim \sqrt{2 \log r}$.

Theorem 4.3 (Barrier-aware Attention load)**.** *Under the assumptions of Lemma 4.1 with* $\nu > 0$*, for every fixed integer* $r \ge 1$*,*

$$\frac{W_{B,r} - B\theta}{\sqrt{B}\,\nu} \;\Rightarrow\; M_r \quad \text{as } B \to \infty, \tag{6}$$

and

$$\mathbb{E}[W_{B,r}] = B\theta + \sqrt{B}\,\nu\,\kappa_r + o(\sqrt{B}). \tag{7}$$

If $\nu = 0$*, then* $W_{B,r} = B\theta$ *exactly.*

The proof, in Appendix A.2, combines the multivariate central limit theorem with a uniform-integrability argument bounding $\mathbb{E}[(\max_j X_j^{(B)})^2] \le r$. Equation (7) gives the barrier-aware replacement for the mean-field Attention load: $B\theta \quad \rightsquigarrow \quad B\theta + \sqrt{B}\,\nu\,\kappa_r$. The relative synchronization overhead is $(\nu/\theta)(\kappa_r/\sqrt{B})$, which grows sublogarithmically with r via $\kappa_r \sim \sqrt{2 \log r}$ and decays as $B^{-1/2}$.

Monte Carlo validation. We verify the Gaussian approximation under geometric decode lifetimes (Corollary 4.5) with $B = 256$, $\mu_P = 100$, and $\mu_D = 500$ ($\mu_{\text{out}} = 499$). By Corollary 4.5, $\theta = \mu_P + \mu_{\text{out}}$ and $\nu^2 = \sigma_P^2 + \mu_{\text{out}}(\mu_{\text{out}} + 1)$, giving predicted overhead $\sqrt{B}\,\nu\,\kappa_r/(B\theta)$. Table 1 (Appendix A.3) compares this against Monte Carlo estimates (50,000 trials per r): the CLT prediction matches within 0.5% for all $r \in \{2, \ldots, 24\}$. The overhead reaches $\sim$11% at $r = 24$, accounting for the majority of the $\sim$15% throughput gap observed at large r (Section 5). Importantly, after incorporating this correction into (12), the simulation-optimal r^* remains at 8, confirming that the mean-field rule (10) already provides an accurate provisioning recommendation.

4.3 Cycle Time Approximations

We work with two approximations to the expected cycle time $\bar{\tau}(B;r)$. Define

$$\mu_A := \alpha_A B\theta + \beta_A, \qquad \sigma_A := \alpha_A\sqrt{B}\,\nu, \qquad G_{B,r} := \max\{\alpha_C rB + \beta_C,\ \alpha_F rB + \beta_F\}.$$

Mean-field cycle time. Ignoring the cross-worker barrier corresponds to replacing $W_{B,r}$ by its mean-field value $B\theta$:

$$\tau_{\mathrm{mf}}(B;r) := \max\{\mu_A,\ \alpha_C rB + \beta_C,\ \alpha_F rB + \beta_F\}\,. \tag{8}$$

This is the natural deterministic surrogate when stochastic imbalance is fully absorbed by an idealized balancing policy ($\nu_{\mathrm{eff}} \to 0$).

Gaussian cycle time. Retaining the barrier, the leading-order Gaussian approximation to $\bar{\tau}(B;r)$ is

$$\tau_G(B;r) := G_{B,r} + \sigma_A \int_{z_{B,r}}^{\infty} (m - z_{B,r})\, r\phi(m)\Phi(m)^{r-1}\, dm, \qquad z_{B,r} := \frac{G_{B,r} - \mu_A}{\sigma_A}, \tag{9}$$

with $\bar{\tau}(B;r) = \tau_G(B;r) + o(\sqrt{B})$. The integral admits inexpensive numerical quadrature; Appendix A.4 gives the derivation and a closed-form expression for small r.

4.4 Provisioning Rules

We present a closed-form mean-field rule and a barrier-aware refinement.

Theorem 4.4 (Mean-field optimal A/F ratio)**.** *Under the mean-field cycle time* (8)*, the throughput is maximized by evaluating the following candidate ratios:*

$$\boxed{r^*_{\mathrm{mf}} \in \left\{ \min\left\{ \frac{\mu_A - \beta_C}{\alpha_C B}, \frac{\mu_A - \beta_F}{\alpha_F B} \right\},\quad \sqrt{\frac{\beta_C}{\alpha_C B}},\quad \sqrt{\frac{\beta_F}{\alpha_F B}},\quad \frac{\beta_C - \beta_F}{B(\alpha_F - \alpha_C)} \right\}.} \tag{10}$$

Let r^*_{mf} be the candidate in (10) *with the largest* $\mathrm{Thr}_{\mathrm{mf}}(B;r)$*. Then*

$$\mathrm{Throughput}^*_{\mathrm{mf}} = \frac{r^*_{\mathrm{mf}} B}{(r^*_{\mathrm{mf}} + 1)\tau_{\mathrm{mf}}(B; r^*_{\mathrm{mf}})}.$$

The proof is in Appendix A.5. Since communication and FFN both scale with the aggregate batch rB, communication enters through τ_{mf} directly, rather than through a separate θ-dependent balance term. **Barrier-aware refinement.** When the cross-worker barrier is retained, the throughput

$$\mathrm{Thr}_G(B;r) := \frac{rB}{(r+1)\,\tau_G(B;r)} \tag{11}$$

is no longer maximized in closed form, because $\tau_G(B;r)$ depends on r both through the FFN term $\alpha_F rB + \beta_F$ in $G_{B,r}$ and through the order-statistic factor $\kappa_r \sim \sqrt{2\log r}$ inside the integral. The barrier-aware optimal ratio is

$$\boxed{r^*_G \in \arg\max_{r\in\mathcal{R}} \mathrm{Thr}_G(B;r),} \tag{12}$$

where $\mathcal{R}$ is the feasible set of integer fan-ins. This is a one-dimensional analytic optimization evaluable in milliseconds combined with a discrete search over r. In the practically common setting where decode lengths approximately follow a geometric distribution—as observed across multiple production LLM traces (Figure 5 in Appendix A.8)—the moment statistics admit closed-form expressions, yielding the following specialization.

Corollary 4.5 (Geometric decode lifetimes)**.** *If P and D are independent and $D \sim \mathrm{Geom}(p)$ on $\{1,2,\ldots\}$, write $\mu_{\mathrm{out}} := (1-p)/p$ for the expected number of generated tokens. Then*

$$\theta = \mu_P + \mu_{\mathrm{out}}, \qquad \nu^2 = \sigma_P^2 + \mu_{\mathrm{out}}(\mu_{\mathrm{out}} + 1),$$

so the mean-field rule (10) *reduces to the form derived in prior analyses of LLM decoding under geometric assumptions, while* (12) *sharpens it with the synchronization correction.*

Practical recipe. Given hardware parameters $(\alpha_A, \beta_A, \alpha_F, \beta_F, \alpha_C, \beta_C)$ and request traces $(P_i, D_i)_{i=1}^n$: (i) Estimate $\hat{\theta}$ and $\hat{\nu}^2$ from the trace using the nonparametric estimator (Appendix A.6). (ii) Compute r^*_{mf} in closed form via Theorem 4.4 as a fast initial recommendation. (iii) Refine via r^*_G from (12) when cross-worker imbalance is non-negligible.

5 NUMERICAL EXPERIMENTS

To validate our theoretical framework, we develop an AFD simulator and conduct systematic experiments comparing theoretical predictions with simulation results. The source code is available at `https://anonymous.4open.science/r/AF-release-1C11`.

5.1 SIMULATOR DESIGN

The simulator implements a discrete-event, cycle-by-cycle simulation of the rA–1F topology, where each `Batch` object transitions through a six-state finite state machine (Attention $\rightarrow$ A2F transfer $\rightarrow$ Waiting $\rightarrow$ FFN $\rightarrow$ F2A transfer $\rightarrow$ Waiting $\rightarrow$ repeat). To maximize resource utilization, the simulator maintains two batches in flight simultaneously: while one batch is being processed by the shared FFN server, the Attention instances process the other batch in parallel, and vice versa. This interleaved execution allows computation on one batch to overlap with communication and processing of the other, effectively hiding transfer latencies.

5.2 EXPERIMENTAL SETUP

Latency parameters and configuration. We adopt linear latency coefficients calibrated for the DeepSeek-V3 architecture deployed on Huawei Ascend 910C NPUs, obtained via linear regression on real execution traces: $\alpha_A = 0.00165$ cycles/token, $\beta_A = 50$ cycles, $\alpha_F = 0.083$ cycles/request, $\beta_F = 100$ cycles, $\alpha_C = 0.022$ cycles/token, and $\beta_C = 20$ cycles. Appendix B provides the general derivation framework, enabling practitioners to calibrate these parameters for other hardware platforms. Under this configuration, communication latency can be effectively hidden through pipelining ($t_A, t_F > 2t_C$ across operating regimes), so our analysis focuses on balancing Attention and FFN workloads. We fix batch size $B = 256$, expected decode length $\mu_D = 500$ ($\sigma_D^2 = 294500$) tokens, and expected prefill length $\mu_P = 100$ ($\sigma_P^2 = 9900$) tokens. We sweep $r \in \{1, 2, 4, 8, 16, 24, 32\}$ and simulate until $N = 10{,}000$ requests complete per Attention instance. By Theorem 4.4, The theoretical optimal A/F ratio is: $r^*_{\text{mf}} \approx 9.3$.

Evaluation Metrics. We evaluate the following metrics to assess system performance:

- **Stable Throughput per Instance:** To avoid distortion from startup transients and tail effects (e.g., the final requests draining with partially filled batches), we report throughput computed over the first 80% of completed requests. Specifically, we measure $T_{80\%}$ as the time when $\lceil 0.8N \rceil$ requests have completed, and compute $\text{Throughput}^{(80\%)}_{\text{per-inst}} = \frac{1}{r+1} \cdot \frac{\sum_{i=1}^{\lceil 0.8N \rceil} D_i}{T_{80\%}}$, where D_i is the number of output tokens generated for the i-th request.
- **Time Per Output Token (TPOT)**: The average time to generate a single output token per request, defined as $\text{TPOT} = \frac{1}{N} \sum_{i=1}^{N} \frac{T^{(i)}_{\text{decode}}}{D_i}$, where $T^{(i)}_{\text{decode}}$ denotes the total decode time for the i-th request.
- **Idle Ratios**: To quantify resource utilization, we measure the fraction of time that each component remains idle waiting for synchronization: $\eta_A = \frac{1}{r} \sum_{i=1}^{r} \frac{t^{(i)}_{\text{idle},A}}{T_{\text{total}}}$, $\eta_F = \frac{t_{\text{idle},F}}{T_{\text{total}}}$, where $t^{(i)}_{\text{idle},A}$ and $t_{\text{idle},F}$ denote the total idle waiting time of the i-th Attention instance and the shared FFN instance, respectively. Lower values indicate better utilization; in an ideally balanced system, both $\eta_A, \eta_F \rightarrow 0$.

5.3 RESULTS

Figure 3 shows per-instance throughput, TPOT, and idle ratios as functions of r. The theoretical optimal $r^*_{\text{mf}} \approx 9.3$ closely matches the simulation-optimal ratio. Moreover, the theoretical throughput curve tracks the simulation results well across all configurations, validating our analytical framework. A residual gap of up to $\sim 15\%$ between the mean-field prediction and simulation emerges at large r, consistent with the barrier-aware analysis in Section 4.2: the synchronization overhead $(\nu/\theta)(\kappa_r/\sqrt{B})$ grows with r, reaching $\sim 11\%$ at $r = 24$ (Table 1), which accounts for the majority of the observed discrepancy. The barrier-aware refinement (12) closes this gap in throughput estimation; in the present configuration, both rules agree on the simulation-optimal r^*, indicating

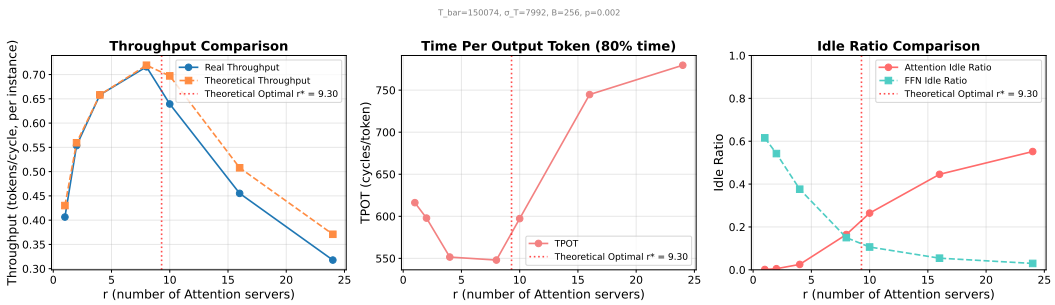
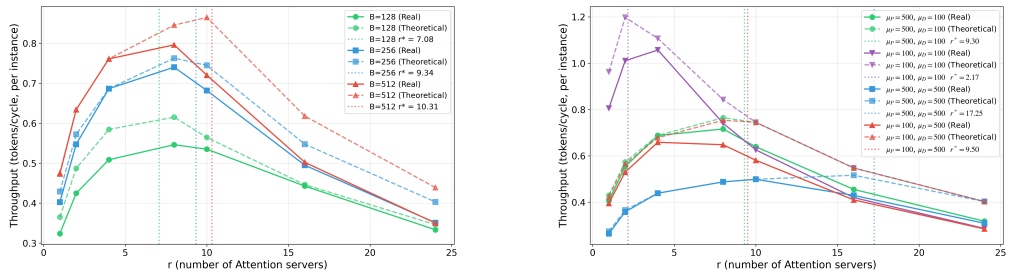


Figure 3: Per-instance throughput, TPOT, and idle ratios as functions of A/F ratio r with $B = 256$, $\mu_D = 500$, $\mu_P = 100$. Left: throughput increases with r until reaching the optimal point $r^* \approx 9.3$, after which FFN becomes saturated and throughput per instance decreases. Middle: TPOT decreases and then increases, with the minimum near r^*. Right: the crossover point where $\eta_A \approx \eta_F$ indicates the balanced configuration.



(a) Impact of batch size B

(b) Impact of μ_P, μ_D

Figure 4: Ablation studies on per-instance throughput and optimal A/F ratio r^*. Solid lines denote simulation results; dashed lines denote theoretical predictions. Vertical dotted lines indicate the theoretical optimal r^* for each configuration.

that the closed-form mean-field rule already provides accurate provisioning under typical operating conditions.

Figure 3 (right) presents the idle ratios for Attention (η_A) and FFN (η_F) instances. When r is small, FFN completes quickly and idles while waiting for Attention outputs ($\eta_F > 60\%$ at $r = 1$). As r increases, more Attention instances feed the shared FFN, improving its utilization.

5.4 Ablation Studies on Workload and Configuration Parameters

We now investigate how the optimal A/F ratio r^* varies with key system parameters: microbatch size B, decode length distribution parameter p, and prefill length distribution parameter q.

Ablation on Batch Size B. Figure 4a compares throughput across three batch sizes ($B \in \{128, 256, 512\}$) with corresponding theoretical optimal ratios $r^* = \{7.08, 9.34, 10.31\}$. The results demonstrate that larger batch sizes achieve higher peak throughput: batching amortizes fixed overhead costs across more requests, thereby improving hardware utilization. Meanwhile, the optimal r^* increases moderately with B, as larger batches require more Attention instances to fully saturate the shared FFN server.

Ablation on Workload Distribution. Figure 4b examines the impact of prefill and decode length distribution. The results reveals that the optimal r^* scales with total context length. Both longer prefills (μ_P) and longer decode sequences (μ_D) increase the total token load in the KV cache, placing greater computational pressure on the Attention side. Consequently, more Attention instances are required to balance the shared FFN server.

6 AI USE STATEMENT

Generative AI tools were used solely to polish the writing and improve the readability of the manuscript. All research ideas, experimental design, mathematical derivations, theoretical analysis, and interpretation of results were carried out by the human authors.

REFERENCES

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav 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 24)*, pp. 117–134, 2024.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Lequn Chen. Dissecting batching effects in GPT inference. Blog post, May 2023. URL `https://le.qun.ch/en/blog/2023/05/13/transformer-batching/`.

Zixi Chen, Tianci Bu, Chendong Song, Xin Lu, Yinyu Ye, and Zijie Zhou. A universal load balancing principle and its application to large language model serving. *arXiv preprint arXiv:2601.17855*, 2026.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Jonah Ekelund, Stefano Markidis, and Ivy Peng. Boosting performance of iterative applications on gpus: Kernel batching with cuda graphs. In *2025 33rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 70–77. IEEE, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph 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*, pp. 611–626, 2023.

Qingyuan Li, Bo Zhang, Liang Ye, Yifan Zhang, Wei Wu, Yerui Sun, Lin Ma, and Yuchen Xie. Flash communication: Reducing tensor parallelization bottleneck for fast large language model inference. *arXiv preprint arXiv:2412.04964*, 2024.

Pierre Lienhart. LLM inference series: 5. dissecting model performance. Medium, March 2024. URL `https://medium.com/@plienhar/llm-inference-series-5-dissecting-model-performance-6144aa93168f`.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–15, 2021.

OpenAI. GPT-4 technical report. arxiv 2303.08774. 2(5), 2023.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative llm inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 118–132. IEEE, 2024.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Bin Wang, Bojun Wang, Changyi Wan, Guanzhe Huang, Hanpeng Hu, Haonan Jia, Hao Nie, Mingliang Li, Nuo Chen, Siyu Chen, et al. Step-3 is large yet affordable: Model-system co-design for cost-effective decoding. *arXiv preprint arXiv:2507.19427*, 2025.

Guan Wang, Sijie Cheng, Xianyuan Zhan, Xiangang Li, Sen Song, and Yang Liu. Openchat: Advancing open-source language models with mixed-quality data. *arXiv preprint arXiv:2309.11235*, 2023.

Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Zhenheng Tang, Xin He, Rui Guo, Xin Wang, Qiang Wang, Amelie Chi Zhou, et al. Burstgpt: A real-world workload dataset to optimize llm serving systems. *arXiv preprint arXiv:2401.17644*, 2024.

Xiuying Wei, Skander Moalla, Razvan Pascanu, and Caglar Gulcehre. Building on efficient foundations: Effective training of llms with structured feedforward layers. *Advances in Neural Information Processing Systems*, 37:4689–4717, 2024.

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.

Zhexiang Zhang, Ye Wang, Xiangyu Wang, Yumiao Zhao, Jingzhe Jiang, Qizhen Weng, Shaohuai Shi, Yin Chen, and Minchen Yu. Janus: Disaggregating attention and experts for scalable moe inference. *arXiv preprint arXiv:2512.13525*, 2025.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatgpt interaction logs in the wild. *arXiv preprint arXiv:2405.01470*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P Xing, et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023.

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 24)*, pp. 193–210, 2024.

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.

Pengfei Zuo, Huimin Lin, Junbo Deng, Nan Zou, Xingkun Yang, Yingyu Diao, Weifeng Gao, Ke Xu, Zhangyu Chen, Shirui Lu, et al. Serving large language models on huawei cloudmatrix384. *arXiv preprint arXiv:2506.12708*, 2025.

A Supplementary Materials for Section 4

A.1 Proof of Lemma 4.1

We use the discrete-time renewal-reward theorem applied to one decode slot. Each request constitutes one renewal cycle. Cycle n has length D_n steps, during which the slot visits ages $a = 0, 1, \ldots, D_n - 1$. For any measurable function $g(P, a)$ with $\mathbb{E}[\sum_{a=0}^{D-1} |g(P, a)|] < \infty$, the renewal-reward theorem gives the long-run time average

$$\mathbb{E}_\pi[g(P, A)] = \frac{\mathbb{E}\left[\sum_{a=0}^{D-1} g(P, a)\right]}{\mathbb{E}[D]}, \tag{13}$$

where A denotes the stationary age and the expectation on the right is taken over a single cycle.

First moment. Take $g(P, a) = P + a$. Using $\sum_{a=0}^{D-1}(P + a) = DP + D(D-1)/2$,

$$\theta = \mathbb{E}_\pi[P + A] = \frac{\mathbb{E}[DP + D(D-1)/2]}{\mathbb{E}[D]},$$

which proves the first identity in (3). Condition $\mathbb{E}[D] < \infty$ together with (2) ensures the integrability required by (13).

Second moment. Take $g(P, a) = (P + a)^2$. Expanding and using $\sum_{a=0}^{D-1} 1 = D$, $\sum_{a=0}^{D-1} a = D(D-1)/2$, and $\sum_{a=0}^{D-1} a^2 = D(D-1)(2D-1)/6$,

$$\sum_{a=0}^{D-1}(P + a)^2 = DP^2 + PD(D-1) + \frac{D(D-1)(2D-1)}{6}.$$

Substituting into (13) proves the second identity in (3), and $\nu^2 = \mathbb{E}[Y^2] - \theta^2$.

Independent case. If $P \perp\!\!\!\perp D$, then $\mathbb{E}[DP]/\mathbb{E}[D] = \mu_P$ and

$$\frac{\mathbb{E}[D(D-1)]}{2\mathbb{E}[D]} = \frac{\mu_D^2 + \sigma_D^2 - \mu_D}{2\mu_D} = \frac{\mu_D - 1}{2} + \frac{\sigma_D^2}{2\mu_D},$$

which proves (4). Without independence, $\mathbb{E}[DP]/\mathbb{E}[D] = \mu_P + \mathrm{Cov}(P, D)/\mu_D$, giving the additional covariance term. □

A.2 Proof of Theorem 4.3

Assume $\nu > 0$ (the case $\nu = 0$ is immediate, since $Y_{j,b} = \theta$ a.s. implies $W_{B,r} = B\theta$ exactly). Define

$$X_j^{(B)} := \frac{T_j^{(B)} - B\theta}{\sqrt{B}\,\nu}, \qquad j = 1, \ldots, r.$$

Each $T_j^{(B)} = \sum_{b=1}^{B} Y_{j,b}$ is a sum of B i.i.d. random variables with mean θ and variance $\nu^2 < \infty$ (by Lemma 4.1). By the classical central limit theorem, $X_j^{(B)} \Rightarrow Z_j$ with $Z_j \sim \mathcal{N}(0, 1)$, and the multivariate CLT gives

$$(X_1^{(B)}, \ldots, X_r^{(B)}) \;\Rightarrow\; (Z_1, \ldots, Z_r),$$

where $Z_1, \ldots, Z_r$ are independent because the workers are independent. The maximum function $h(x_1, \ldots, x_r) = \max_j x_j$ is continuous, so the continuous mapping theorem yields $\max_j X_j^{(B)} \Rightarrow M_r$. Combined with the identity $W_{B,r} = B\theta + \sqrt{B}\,\nu \max_j X_j^{(B)}$, this proves (6).

For (7), observe that $(\max_j X_j^{(B)})^2 \le \sum_{j=1}^{r} (X_j^{(B)})^2$, so

$$\mathbb{E}\left[(\max_j X_j^{(B)})^2\right] \le \sum_{j=1}^{r} \mathbb{E}\left[(X_j^{(B)})^2\right] = r,$$

which establishes uniform integrability of $\{\max_j X_j^{(B)}\}_{B \ge 1}$. Combining uniform integrability with weak convergence yields $\mathbb{E}[\max_j X_j^{(B)}] \to \kappa_r$, and taking expectations in $W_{B,r} = B\theta + \sqrt{B}\,\nu \max_j X_j^{(B)}$ proves (7). □

A.3 Monte Carlo Validation of the Barrier Approximation

We validate Theorem 4.3 by comparing the predicted relative synchronization overhead $\kappa_r s/m$ against Monte Carlo (MC) estimates. The experiment uses geometric decode lifetimes (Corollary 4.5) with $B = 256$, $\mu_P = 100$, $\mu_D = 500$. Each worker's token load sums B independent slot loads, giving $T_j \approx \mathcal{N}(m, s^2)$ with $m = B(\mu_P + \mu_D)$ and $s = \sqrt{B(\sigma_P^2 + \mu_D(\mu_D + 1))}$. MC estimates use $50{,}000$ independent trials per value of r.

Table 1: Relative synchronization overhead: Monte Carlo vs. CLT prediction ($B = 256$, $\mu_P = 100$, $\mu_D = 500$).

r	MC overhead	CLT prediction
2	2.98%	3.00%
4	5.52%	5.47%
8	7.74%	7.57%
12	8.88%	8.66%
16	9.66%	9.39%
24	11.37%	11.01%

The CLT prediction matches the MC estimate within 0.5% across all tested values of r, confirming the accuracy of the Gaussian order-statistic approximation.

A.4 Derivation of the Gaussian Cycle Time (9)

We show that $\bar{\tau}(B; r) = \tau_G(B; r) + o(\sqrt{B})$, where τ_G is defined in (9).

Let $X_B := \max_j X_j^{(B)}$, so that $\alpha_A W_{B,r} + \beta_A = \mu_A + \sigma_A X_B$. The identity $\max\{a, b\} = b + (a-b)_+$ gives

$$\max\{\mu_A + \sigma_A X_B,\ G_{B,r}\} = G_{B,r} + \sigma_A (X_B - z_{B,r})_+,$$

and similarly with X_B replaced by M_r. Taking expectations,

$$\bar{\tau}(B; r) = G_{B,r} + \sigma_A \mathbb{E}[(X_B - z_{B,r})_+]. \tag{14}$$

We claim $\mathbb{E}[(X_B - z_{B,r})_+] - \mathbb{E}[(M_r - z_{B,r})_+] \to 0$ as $B \to \infty$.

Case 1: $z_{B,r}$ has a finite subsequential limit z. The map $x \mapsto (x - z)_+$ is continuous with linear growth; weak convergence $X_B \Rightarrow M_r$ together with uniform L^2-boundedness ($\sup_B \mathbb{E}[X_B^2] \le r$) yields $\mathbb{E}[(X_B - z)_+] \to \mathbb{E}[(M_r - z)_+]$, and Lipschitz continuity in z extends the conclusion to $z_{B,r} \to z$.

Case 2: $z_{B,r} \to +\infty$. Cauchy–Schwarz and Markov's inequality give $\mathbb{E}[(X_B - z_{B,r})_+] \le \mathbb{E}[X_B^2]/z_{B,r} \to 0$, and similarly for M_r.

Case 3: $z_{B,r} \to -\infty$. Write $(X_B - z_{B,r})_+ = (X_B - z_{B,r}) - (X_B - z_{B,r})_-$. The negative part satisfies $(X_B - z_{B,r})_- \le |X_B| \mathbf{1}\{|X_B| > |z_{B,r}|\}$, whose expectation tends to zero by uniform L^2-boundedness; the same bound applies to M_r, and $\mathbb{E}[X_B] \to \kappa_r$.

In all three cases, the difference vanishes. Substituting into (14) and using the density $f_{M_r}(m) = r\phi(m)\Phi(m)^{r-1}$ to evaluate $\mathbb{E}[(M_r - z_{B,r})_+]$ as the integral in (9) completes the proof. □

Closed form for small r. For $r = 1$, $M_1 \sim \mathcal{N}(0, 1)$ and the integral in (9) reduces to $\phi(z) - z(1 - \Phi(z))$ at $z = z_{B,1}$. For $r = 2$, the integral admits a closed form in terms of ϕ, Φ, and Owen's T-function. For $r \ge 3$, numerical quadrature is straightforward.

A.5 Proof of Theorem 4.4

Recall

$$G_{B,r} = \max\{\alpha_C B r + \beta_C, \alpha_F B r + \beta_F\}, \qquad \tau_{\mathrm{mf}}(B; r) = \max\{\mu_A, G_{B,r}\}.$$

When $G_{B,r} \le \mu_A$,

$$\mathrm{Thr}_{\mathrm{mf}}(B;r) = \frac{B}{\mu_A}\frac{r}{r+1},$$

which is increasing in r. This region ends at

$$\min\left\{\frac{\mu_A-\beta_C}{\alpha_C B}, \frac{\mu_A-\beta_F}{\alpha_F B}\right\}.$$

It remains to consider the region where $G_{B,r}$ is active. On any smooth branch $G_{B,r} = \alpha_i B r + \beta_i$, $i \in \{C, F\}$,

$$\frac{d}{dr}\log\frac{rB}{(r+1)(\alpha_i B r+\beta_i)} = \frac{\beta_i - \alpha_i B r^2}{r(r+1)(\alpha_i B r+\beta_i)}.$$

Thus the only smooth stationary points are

$$\sqrt{\frac{\beta_C}{\alpha_C B}}, \qquad \sqrt{\frac{\beta_F}{\alpha_F B}}.$$

The only nonsmooth point is the crossing of the communication and FFN latencies,

$$\frac{\beta_C-\beta_F}{B(\alpha_F-\alpha_C)}.$$

Therefore the global optimum must be one of the candidates in (10); choosing the one with the largest $\mathrm{Thr}_{\mathrm{mf}}(B;r)$ gives r^*_{mf}. Substitution gives the stated optimal throughput. □

A.6 Trace-Based Nonparametric Estimator

Given a request trace $(P_i, D_i)_{i=1}^n$, the framework requires no parametric model. Estimate θ and the second moment of Y by

$$\hat\theta = \frac{\sum_{i=1}^n \left[D_i P_i + D_i(D_i-1)/2\right]}{\sum_{i=1}^n D_i}, \tag{15}$$

$$\hat q = \frac{\sum_{i=1}^n \left[D_i P_i^2 + P_i D_i(D_i-1) + D_i(D_i-1)(2D_i-1)/6\right]}{\sum_{i=1}^n D_i}, \tag{16}$$

and set $\hat\nu^2 = \hat q - \hat\theta^2$. Both estimators are ratios of i.i.d. sums and are strongly consistent under the moment conditions of Lemma 4.1. A standard delta-method argument gives $\sqrt{n}$-asymptotic normality under the additional condition that the corresponding cycle-reward variables have finite second moments. Plugging $(\hat\theta, \hat\nu)$ into Theorems 4.4 and the barrier-aware rule (12) yields the empirical provisioning rule, computable directly from serving logs.

A.7 Heavy-Tailed Decode Lifetimes

The finite-variance Theorem 4.3 requires $\nu^2 < \infty$. Because the stationary age distribution is length-biased, this is stronger than finite variance of D. If $\mathbb{P}(D > x) \sim C x^{-\alpha}$, the stationary age A has tail

$$\mathbb{P}(A > x) = \frac{\sum_{a>x}\mathbb{P}(D>a)}{\mathbb{E}[D]} \sim \frac{C}{\mathbb{E}[D](\alpha-1)}\, x^{-(\alpha-1)},$$

shifting the tail exponent from α to $\alpha - 1$. Consequently:

- $\alpha > 3$: $\nu^2 < \infty$, and Theorem 4.3 applies.
- $2 < \alpha \le 3$: $\theta < \infty$ but $\nu^2 = \infty$. The Gaussian $\sqrt{B}$ correction is replaced by stable-law fluctuations; if Y lies in the domain of attraction of a γ-stable law with $\gamma = \alpha - 1$, then $W_{B,r} = B\theta + B^{1/\gamma}\,\mathbb{E}[\max_{1\le j\le r} S_j] + o(B^{1/\gamma})$ for i.i.d. γ-stable S_j.
- $\alpha \le 2$: θ may be infinite; the mean-field token load is undefined.

In practice, serving systems impose maximum context and generation lengths, restoring all moments, so condition (2) is satisfied.

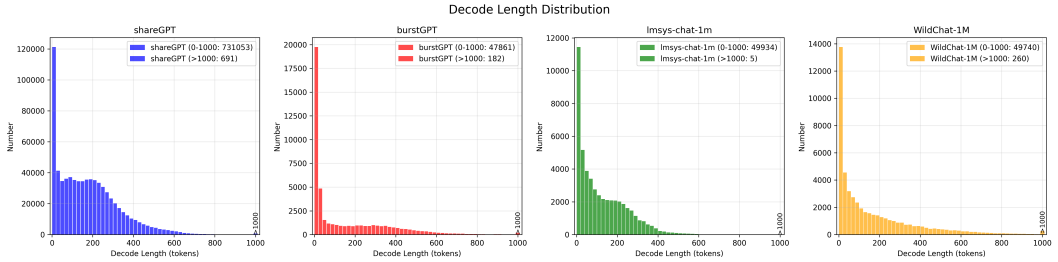


Figure 5: Empirical distributions of decode lengths from production LLM traces Wang et al. (2023; 2024); Zheng et al. (2023); Zhao et al. (2024). Decode lengths exhibit a geometric (discrete-exponential) pattern.

A.8 Evidence of Approximately Geometric Decode Length

Figure 5 shows empirical distributions of decode lengths from production LLM traces Wang et al. (2023; 2024); Zheng et al. (2023); Zhao et al. (2024).

B Supplementary Materials for Section 5

This appendix presents the methodology for deriving the latency model parameters $\alpha_A, \beta_A, \alpha_F, \beta_F, \alpha_C, \beta_C$ used in the main text. We demonstrate the derivation process using the DeepSeek-V3 architecture, but the methodology generalizes to other models and hardware platforms.

Note on hardware parameters. Due to confidentiality constraints, we cannot disclose the explicit Huawei Ascend 910C hardware specifications used in our experiments. Instead, we present the derivation framework using symbolic hardware parameters (summarized in Table 2), enabling practitioners to apply this methodology to their own hardware platforms. The final parameter values in Table 3 were obtained via linear regression on real execution traces over Huawei Ascend 910C.

Table 2: Hardware parameters (platform-specific, values not disclosed)

Symbol	Description
π_{peak}	Peak INT8 compute throughput (TFLOPS)
β_{HBM}	Peak HBM memory bandwidth (TB/s)
η_{mem}	Effective memory bandwidth utilization ratio
η_{compute}	Effective compute utilization ratio
β_{intra}	Intra-node interconnect bandwidth (GB/s)
β_{inter}	Inter-node network bandwidth (GB/s)
N_{cards}	Number of accelerator cards in deployment
$N_{\text{expert/card}}$	Number of experts per card

B.1 Model Configuration

We base our derivations on the DeepSeek-V3 architecture. The model has a hidden size of $H = 7168$ and uses Multi-head Latent Attention (MLA) with a compressed KV cache dimension of $(d_c + d_{\text{rope}}) = 512 + 64 = 576$. For the MoE layers, the expert intermediate dimension is $d_{\text{expert}} = 2048$, with a total of $N_{\text{expert}} = 256$ experts across the system, where each token is routed to $k = 8$ experts. We use Multi-Token Prediction (MTP) with depth 1.

B.2 MLA Attention (Memory-Bound)

During decoding, Attention computation is *memory-bound*, dominated by reading the compressed KV cache from HBM. The compute cost scales with total token load $T = \sum_{b=1}^{B}(s_b + i_b)$.

B.2.1 DERIVATION

Data volume per token. With KV compression dimension $(d_c + d_{\text{rope}}) = 576$ and BF16 precision (2 bytes per element):

$$V_{\text{token}} = (d_c + d_{\text{rope}}) \times 2 = 576 \times 2 = 1152 \text{ bytes}. \tag{17}$$

Effective HBM bandwidth. Accounting for non-coalesced memory access patterns, cache effects, and memory controller overhead, the effective bandwidth is:

$$\beta_{\text{eff}} = \beta_{\text{HBM}} \times \eta_{\text{mem}}. \tag{18}$$

Time per token (slope). The per-token memory access time gives the slope parameter:

$$\alpha_A = \frac{V_{\text{token}}}{\beta_{\text{eff}}} = \frac{(d_c + d_{\text{rope}}) \times 2}{\beta_{\text{HBM}} \times \eta_{\text{mem}}}. \tag{19}$$

Fixed overhead (intercept). The intercept β_A captures the other projections in the attention block, including Query/Key/Value/Output projections in memory-bound status as well as vector operations such as RMSNorm or RoPE. This parameter is best determined empirically via profiling or linear regression on execution traces.

B.3 MOE FFN (COMPUTE-BOUND)

The FFN computation is *compute-bound* when batch size is sufficiently large. Let B_e denote the batch size per expert.

B.3.1 DERIVATION

FLOPs per expert. For SwiGLU activation with 3 weight matrices (quantized gate, up, down projections):

$$\text{FLOPs per expert} = 6 \times H \times d_{\text{expert}} \times B_e = 6 \times 7168 \times 2048 \times B_e \approx 8.81 \times 10^7 \times B_e. \tag{20}$$

Effective compute throughput. For MoE workloads with dynamic routing and load imbalance, the effective throughput is:

$$\pi_{\text{eff}} = \pi_{\text{peak}} \times \eta_{\text{compute}}. \tag{21}$$

Compute time per expert.

$$t_{\text{expert}}(B_e) = \frac{6 \times H \times d_{\text{expert}} \times B_e}{\pi_{\text{eff}}} = \frac{6 \times H \times d_{\text{expert}}}{\pi_{\text{peak}} \times \eta_{\text{compute}}} \cdot B_e. \tag{22}$$

Total FFN time per card. With $N_{\text{expert/card}}$ experts per card:

$$t_{\text{FFN,card}}(B_e) = N_{\text{expert/card}} \times t_{\text{expert}}(B_e) + \beta_{\text{overhead}}. \tag{23}$$

Batch size mapping. The relationship between global batch B_F and batch per expert B_e:

$$B_e = \frac{B_F \times k \times (1 + \text{MTP depth})}{N_{\text{expert}}} = \frac{B_F \times 8 \times 2}{256} = \frac{B_F}{16}. \tag{24}$$

Express in terms of global batch. Substituting the batch size mapping, the FFN latency takes the form:

$$t_F(B_F) = \alpha_F \cdot B_F + \beta_F, \tag{25}$$

where:

$$\alpha_F = \frac{N_{\text{expert/card}} \times 6 \times H \times d_{\text{expert}}}{\pi_{\text{peak}} \times \eta_{\text{compute}}} \times \frac{k \times (1 + \text{MTP depth})}{N_{\text{expert}}}. \tag{26}$$

B.4 COMMUNICATION

Communication involves transferring activations between Attention and FFN instances via high-speed interconnects.

B.4.1 DERIVATION

Data volume per batch. For a batch of B_e tokens routed to experts, the activation transfer includes input (factor of 1 with INT8 format) and output (factor of 2 with BF16 format) tokens:

$$V_{\text{comm}}(B_e) = 3 \times H \times B_e = 3 \times 7168 \times B_e = 21504 \times B_e \text{ bytes}. \tag{27}$$

Effective network bandwidth. For the AFD topology spanning both intra-node and inter-node communication, the effective aggregated bandwidth depends on the network topology and routing:

$$\beta_{\text{net}} = f(\beta_{\text{intra}}, \beta_{\text{inter}}, \text{topology}), \tag{28}$$

where the function f depends on the specific deployment configuration.

Transfer time.

$$t_{\text{comm}}(B_e) = \frac{N_{expert/card} V_{\text{comm}}(B_e)}{\beta_{\text{net}}} + \beta_C = \frac{3 \times H}{\beta_{\text{net}}} \cdot B_e + \beta_C. \tag{29}$$

Convert to per-token formulation. With batch size mapping $B_e = B_F/16$, the communication latency becomes:

$$t_C(rB) = \alpha_C \cdot rB + \beta_C, \tag{30}$$

where:

$$\alpha_C = N_{expert/card} \times \frac{3 \times H}{\beta_{\text{net}}} \times \frac{k \times (1 + \text{MTP depth})}{N_{\text{expert}}}. \tag{31}$$

B.5 SUMMARY

The derivations above provide a general framework for computing latency parameters from hardware specifications and model architecture. The key relationships are:

- **Attention (memory-bound):** $\alpha_A \propto \dfrac{d_c + d_{\text{rope}}}{\beta_{\text{HBM}} \times \eta_{\text{mem}}}$
- **FFN (compute-bound):** $\alpha_F \propto \dfrac{H \times d_{\text{expert}}}{\pi_{\text{peak}} \times \eta_{\text{compute}}}$
- **Communication:** $\alpha_C \propto \dfrac{H}{\beta_{\text{net}}}$

Table 3 presents the parameter values used in our experiments. These values were obtained via linear regression on real execution traces collected from Huawei Ascend 910C NPUs, which provides empirical validation of the linear latency models while accounting for system-level effects not captured in first-principles analysis.

We visualize the derived latency models in Figure 6, which illustrates how Attention, FFN, and communication latencies scale with their respective input sizes under the parameters in Table 3.

C COMPUTE RESOURCES.

All simulator experiments reported in Figures 3–4 and Appendix A.3 are CPU-only and do not require access to NPUs or GPUs once the latency coefficients are fixed. We ran the simulator on [CPU model] with [number] CPU cores and [RAM] memory. A single fan-in sweep for Figure 3 takes approximately 15 minutes, the batch-size and workload ablations in Figure 4 take approximately 20 minutes in total, and reproducing all reported simulator figures takes approximately 3 CPU-hours. The only accelerator-dependent component is the offline latency calibration used to obtain Table 3. These calibration traces were collected on Huawei Ascend 910C NPUs; raw profiling traces and detailed hardware parameters cannot be released due to confidentiality constraints, so reproduction of the paper's simulator results uses the released fitted coefficients.

Table 3: Latency parameters obtained via linear regression on execution traces (Huawei Ascend 910C)

Equation	Symbol	Value	Unit
Eq. (2)	α_A	1.65×10^{-3}	cycles/token
	β_A	50	cycles
Eq. (3)	α_F	0.083	cycles/request
	β_F	100	cycles
Eq. (4)	α_C	0.022	cycles/token
	β_C	20	cycles

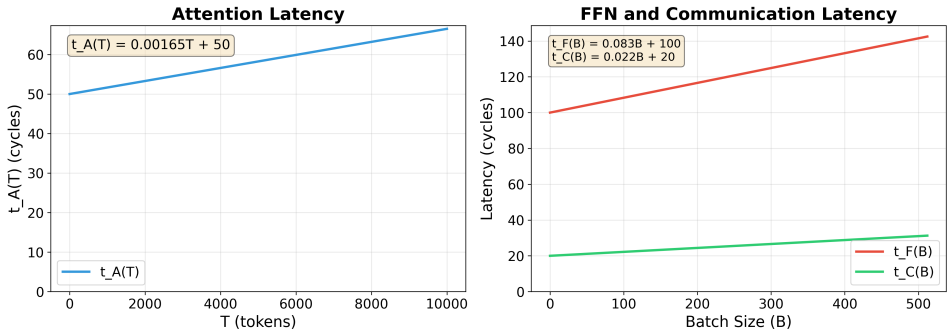


Figure 6: Visualization of the latency models. Left: Attention latency $t_A(T)$ scales linearly with total token load T. Right: FFN latency $t_F(B)$ and communication latency $t_C(rB)$ as functions of batch size B.