ICML 2026
Seoul · July 6–11, 2026
Main Conference Poster

A Queueing-Theoretic Framework for Stability Analysis of LLM Inference with KV Cache Memory Constraints

Chengyi Nie[1] **Nian Si**[2] **Zijie Zhou**[2]
[1] *Department of Electrical and Computer Engineering, Stony Brook University* [2] *Department of Industrial Engineering and Decision Analytics, HKUST*

Track:
Theory of LLM Inference
Keywords:
LLM · KV Cache · Queueing · Stability

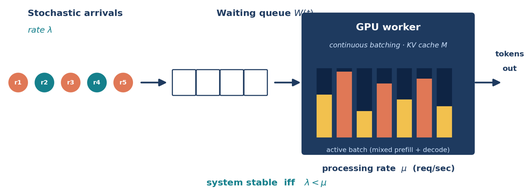
TL;DR

LLM inference at scale is bottlenecked by **BOTH compute AND KV-cache memory**. We build the **first queueing model that captures memory**, derive **closed-form stability conditions**, and validate on real A100 GPUs with GAP < 10%.

Central Question

Given a workload and GPU memory M, how many GPUs are needed to keep an LLM service stable?

LLM Inference as a Queue



Two phases per request: **prefill** (chunked, memory grows by ŝ each step) then **decode** (memory grows by 1 token per step). The GPU runs a mixed batch under a hard memory cap M.

Our Contributions

1. **Modeling** — First queueing model for LLM inference that explicitly captures both compute AND GPU memory (KV cache) constraints — generalizing prior compute-only formulations.
2. **Theory** — Rigorous stability & instability conditions via a Lyapunov / Foster argument adapted to the piecewise-linear KV growth.
3. **Validation** — Extensive real A100 experiments under single- and multi-GPU deployments, plus LongBench v2. Predictions agree to within 10%.

Why It Matters in Practice

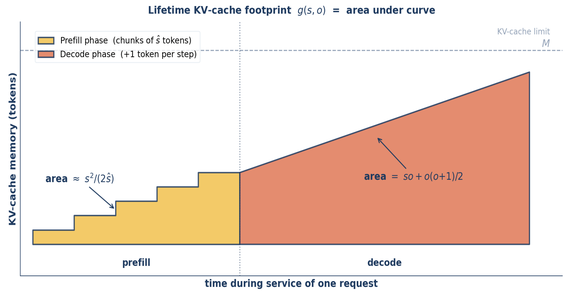
- ✓ **Right-size clusters** — Pick N = ⌈λ/μ⌉ GPUs directly from the workload distribution — no costly trial-and-error rollouts.
- ✓ **Avoid OOM at peak** — The memory-aware rate μ catches regimes that compute-only models silently misjudge as feasible.
- ✓ **Adapt to drift** — Plug a fresh estimate of p(s,o) into μ to autoscale as the workload mix shifts in production.

ICML 2026 · Full paper + proofs in supplementary

Memory Footprint of a Request

For input s, output o, and chunk size ŝ:

$$g(s,o) = \frac{(1+s/\hat{s})\,s \;+\; 2so \;+\; o(o+1)}{2}$$



Key insight: the lifetime **area under the memory curve** is what each request "costs" the GPU. Compressing the trajectory into the scalar g(s, o) makes Lyapunov analysis tractable.

Why Constant Batch Time?

Near saturation, **attention compute dominates**; chunked prefill bounds per-batch work. Empirically **80% of batches share the same execution time** → we model b̄ as constant.

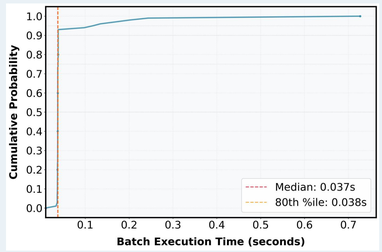
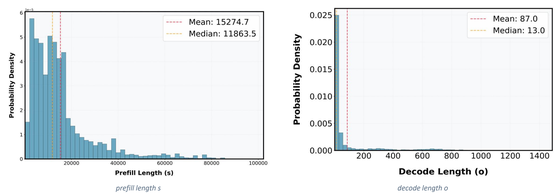


Fig. CDF of batch execution time (Llama-3-8B · A100 · PD 1:1). Median ≈ 80th %ile.

Heavy-Tailed Real Workloads

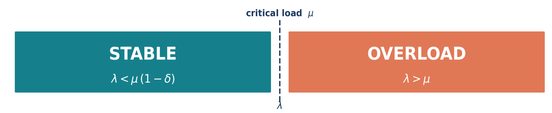
LongBench v2: 503 long-context queries — prefill is heavy-tailed, decode is heavier-tailed. Mean ≫ median in both → modelling the full joint p(s,o) is essential.



Main Result: Stability Condition

Per-GPU service rate

$$\mu = \frac{M}{\bar{b}\cdot \mathbb{E}_{(s,o)\sim p(s,o)}[\,g(s,o)\,]}$$



THEOREM 1 · Overload
If **λ > μ,** the queue grows to infinity under ANY scheduling policy.

THEOREM 2 · Stability
If **λ < μ(1−δ),** with δ = esssup(s+o)/M ≪ 1, any work-conserving + continuous-batching policy is positive recurrent.

Proof Sketch · Lyapunov Drift

Let V(t) = total outstanding KV-cache demand ("area still to erase"). Per slot:

- Saturated GPU erases ≥ M(1−δ) per slot
- Arrivals add λb̄ · 𝔼[g(s,o)] in expectation

$$\mathbb{E}[\,V(t+1) - V(t) \mid \text{state}_t\,] \;\le\; -M(1-\delta) \;+\; \lambda \bar{b}\,\mathbb{E}[g(s,o)]$$

⇒ Negative drift outside a compact set when **λ < μ(1−δ)**. Foster–Lyapunov ⇒ stability. □

GPU Provisioning Rule

$$N_{\text{GPU}} = \lceil \lambda / \mu \rceil$$

Once λ and μ are estimated, **right-size the cluster in one line.**

Worked Example · PD 1:1, Llama-3-8B on A100

GPU memory M	Batch time b̄
131 k tokens	**0.037 s**
Theory μ_theory	Measured μ_gpu
3.263 req/s	**3.387 req/s**

Scope · Extensions · Future Work

Tensor parallelism: treat a TP group as one logical worker — formula for μ is unchanged, swap in TP-specific M and b̄.
Pipeline parallelism / PD disaggregation: tandem queues with separate memory budgets — open theoretical direction.
Speculative decoding & MoE: draft acceptance rate and expert-routing skew enter g(·) multiplicatively — straightforward to plug in.
Beyond stationarity: use the per-slot drift bound as a building block for non-stationary λ(t), online autoscaling, and SLA-aware admission control.

Experimental Validation

Testbed: 8 × NVIDIA A100, Meta-Llama-3-8B, vLLM v1, chunked prefill ŝ=512.
Memory: M = 131 k tokens. Batch time b̄ from median of 10k simulated batches.
Workloads: PD ratios 1:1, 2:1, 1:2, a piecewise-stationary mix, and LongBench v2.

Theory Matches Reality (GAP < 10%)

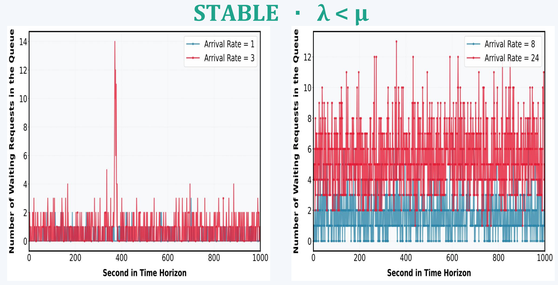
Workload	μ_gpu	μ_theory	GAP
PD 1:1	3.387	3.263	3.66%
PD 2:1	3.650	3.956	8.38%
PD 1:2	2.969	2.902	2.25%
Mixed 2:1 → 1:2	3.137	3.385	7.90%
LongBench v2	0.610	0.561	8.03%
8 × GPU, PD 1:1	**26.710**	**25.808**	**3.38%**

Single- and multi-GPU predictions are within 10% of measurements across every regime — including a heavy-tailed real-world dataset.

Queue Dynamics (single & 8-GPU)

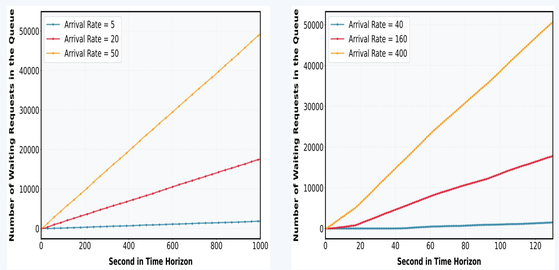
Same conclusion in both scales: when λ < μ the backlog stays bounded; when λ > μ it grows linearly in time.

STABLE · λ < μ



1 GPU · queue stays bounded *8 GPUs · queue stays bounded*

OVERLOAD · λ > μ



1 GPU · linear blow-up *8 GPUs · linear blow-up*

Take-away: *A single closed-form rate μ = M / (b̄ · 𝔼[g(s,o)]) — derived from the LIFETIME KV-cache footprint — predicts when an LLM cluster is stable, accurately enough to size GPU fleets without trial-and-error.*

jerryzhou@ust.hk