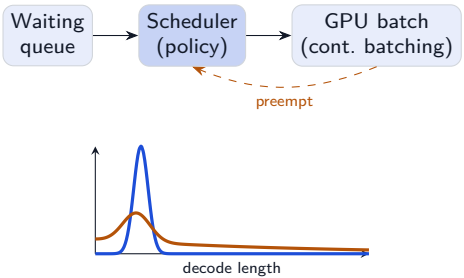
Beyond Prediction

Tail-Aware Scheduling for LLM Inference

Yueying Li, Yuanfan Chen, Jiayang Chen, *et al.*
Cornell University | Microsoft Azure | NVIDIA

Stop predicting decode length – it is hard, and even an oracle can't control the tail. Shape priority softly with distribution-aware queuing theory, manage the KV cache swap jointly, and beat SOTA by 35–50% on P99 TTLT.

Background: what we are scheduling



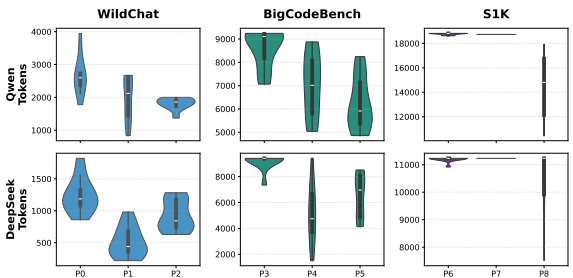
Decode lengths: light-tailed vs heavy-tailed.

- **Schedule:** requests on a GPU replica — prefill (parallel), then decode (1 token/step).
- **Continuous batching:** 1 token per active request per step; admit & retire on the fly.
- **Policy:** admission, priority, preemption (FCFS, SJF, SRPT).

Why it's hard: the best policy depends on the tail regime, and traffic drifts between light-tailed (reasoning) and heavy-tailed (code) — so no static policy wins.

The tail is what hurts, and prediction is brittle

Tail latency (P95 / P99 TTLT) sets the sustainable load of a serving stack. The standard move is to approximate SJF / SRPT by predicting each request's output length.



Same prompt, same model, 20 runs each. Output length varies wildly across chat, code, and reasoning.

Length is unpredictable — so prediction-driven schedulers are unreliable.

Where prior policies fall short

Policy / Work	Heavy-tailed	Light-tailed	No Size Prediction	Preemption Overhead	Tail-Aware by Design	TTLT Evaluated
Shortest Prefix First - SPF (feature in vLLM)	△	△	×	×	×	×
Rank-prediction-based SJF (LTR) (Fu et al., 2024)	✓	△	×	△	×	×
Prediction-based SRPT (TRAIL) (Shahout et al., 2024)	✓	△	×	△	×	✓
FCFS (vLLM)	×	✓	✓	✓	×	△
Skip-Join MLFQ / LAS (Wu et al., 2023)	△	△	✓	△	✓	△
Boost / γ-Boost	×	✓	✓	✓	×	△
Our Work	△	✓	✓	✓	✓	✓

No prior policy fills every column: FCFS protects the tail but blocks; SJF / SRPT (LTR, TRAIL) need length prediction and starve long jobs; pure γ-Boost is tail-aware but memory-oblivious. Ours is the first that is prediction-free, tail-aware, *and* preemption-bounded.

The idea: prediction-free soft priority boosting

Tail-optimal queueing theory replaces hard size ranking with a smooth, continuous score. We adopt the γ-Boost score of **Yu & Scully (2024)**, extended to unknown job sizes by **Harlev et al. (2025)**:

$$\phi_i(t) = a_i - b_\gamma\big(w_i(t)\big), \qquad b_\gamma(w) = \tfrac{1}{\gamma}\log\tfrac{1}{1-e^{-\gamma w}}$$

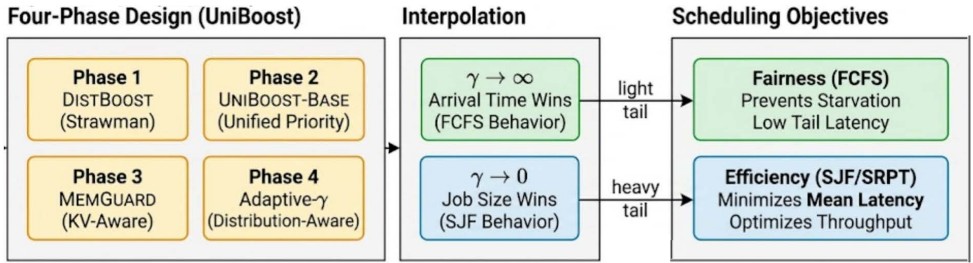
Serve the smallest ϕ. One knob, γ, spans the spectrum:

- $\gamma \to \infty$: behaves like FCFS — protects the tail.
- $\gamma \to 0$: behaves like SJF — chases the mean.

The catch: this theory assumes stateless jobs (M/G/1). LLM requests carry large KV caches, so naive preemption thrashes memory.

Lineage: Nuyens'08 → Nair'10 → Wierman–Zwart'12 → Scully'20 → Grosof'21 → **Yu–Scully'24** → 2025–26 extensions.

UniBoost: a memory-aware co-design



- **Unified priority** over prefill and decode, so new requests are never starved behind long decodes.
- **MemGuard** quantizes priority updates geometrically, capping preemptions at $1 + \lfloor \log_2(S_i/k) \rfloor$ per request.
- **Adaptive** γ tunes itself online from the observed tail.

It beats an oracle with perfect length knowledge

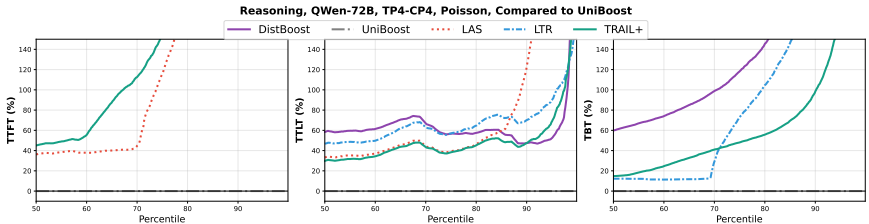
Table 2. Relative performance compared to TRAIL (baseline). Green indicates improvement, yellow indicates minor degradation ($\leq 15\%$), orange indicates moderate degradation ($\leq 50\%$), red indicates severe degradation ($> 50\%$).

	End-to-End Latency			TTFT			TBT			Throughput
Scheduler	Mean	P95	P99	Mean	P95	P99	Mean	P95	P99	tok/s
TRAIL+	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%	+0.0%
SJF	**-11.1%**	**-12.5%**	**+11.2%**	**-74.6%**	**-70.5%**	**-9.7%**	**-5.2%**	**-7.1%**	**-6.3%**	-7.9%
Sarathi	**-12.3%**	**-13.2%**	**-7.6%**	**-49.9%**	**-40.5%**	**-8.1%**	**-6.8%**	**-8.4%**	**-7.9%**	-10.9%
DistBoost	**-14.0%**	**+6.0%**	**+37.1%**	**-22.8%**	**-35.0%**	**-3.4%**	**-14.1%**	**-10.2%**	**+18.3%**	+1.1%
UniBoost	**-19.1%**	**+1.1%**	**+35.1%**	**+52.1%**	**+97.4%**	**+34.0%**	**-25.3%**	**-8.1%**	**+33.8%**	+1.2%

Relative to TRAIL+ (ideal SRPT with perfect decode-length knowledge), Llama-8B mixed workload. Green = better.

UniBoost cuts P99 TTLT ≈35% and P95 TTFT ≈97% without losing throughput. Size-based baselines collapse at the tail — they cannot preempt to reclaim KV memory.

Why the baselines fail at the tail



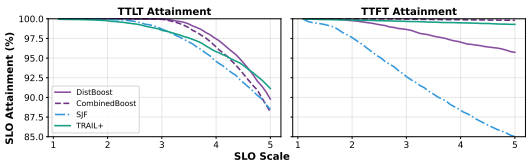
Per-percentile slowdown vs. UniBoost, $100\% \times \left(M_\pi(p)/M_{\text{UniBoost}}(p) - 1\right)$ (QWen-72B, $\rho = 0.99$). UniBoost is the 0% line — *higher = baseline further behind*.

TTFT: prefill deferred behind decode-heavy micro-batches, starving new requests.

TTLT: attained-service inversion — short jobs preempt long ones, thrashing KV.

TBT: over-prioritizing decodes raises concurrency and per-token contention.

What it buys in practice: tighter SLOs



SLO attainment vs. stringency (larger is better).

- **TTFT:** 2.9–8.7× more stringent SLO attainable.
- **TTLT:** 1.7–4.3× more stringent SLO attainable.

Impact: a provider can enforce far tighter latency SLOs than a prediction-based system — higher utilization at the same guarantee.

Stop predicting.

Shape priorities softly, manage the cache jointly,
and the tail takes care of itself.

Drops into vLLM / SGLang continuous batching. **Thank you! | Questions?**