



CONFERENCE

—NORTH AMERICA 2026—

What's the best EP size for rack-scale serving?

GLM-5.3 on GB300 NVL72: where wide EP pays off, and where it stops

Rain Jiang · SGLang · October 2026

One rack, PD disaggregation, from EP4 to EP64

GLM-5.3 in FP8, served 2P1D, without spec decoding

Prefill: 2 trays, with TP = DP = EP = 4,

Decode: scales from 1 tray to all 16 trays, with TP = DP = EP = 4 x per tray.

Workload: 1-token prompts, 1,024 output tokens, closed loop at a fixed number of requests per GPU, 8 rounds per point, FP8 KV cache.

Size	Decode trays	Routers	Cap per rank	KV memory fraction
EP4	1	1	763	0.85
EP8	2	1	2,048	0.85
EP16	4	4	2,560	0.805
EP32	8	4	2,816	0.70
EP64	16	8	2,816	0.70

Lift request cap for server and client with Rust

Serving: Python HTTP server to the embedded Rust server

EP4, 256 per GPU	Python server	Rust server
P50 TTFT	33.9 s	2.7 s
P99 TTFT	89 s	7.3 s
RPS	7.7	17.7
P50 TPOT	92 ms	51 ms
Requests held per rank	decays to about 90	256, the full batch

Benchmarking: Python bench_serving to the Rust client

	Python client	Rust client
Stream parsing	one asyncio thread	16 Tokio threads per process
Saturates at	1,000 to 2,000 streams	not reached
Largest run carried	~2,000 streams	180,224 streams, 16 processes
Symptom when saturated	MBs unread per socket, ranks idle	none

Server numbers: EP4, 1-token prompts, 1,024 output tokens, 1,024 concurrent requests. A tuned Python server (16-token stream interval, 4 tokenizer workers) reached 1.2 s P50 TTFT here but still serves HTTP from rank 0 only; the Rust server gives every DP rank its own listener.

Lift request cap on k8s cluster DNS resolution

Cluster DNS failed 311 of 2,000 concurrent lookups, median 10.5 s, every dns name costs up to 6 queries, and under load the resolver fell over before the model did.

Two fixes. The bench resolves each router once instead of per request, and every pod runs an local dns cache sidecar with prefetch and serve-expired

Failed lookups, 2,000 concurrent

~~311~~ 0

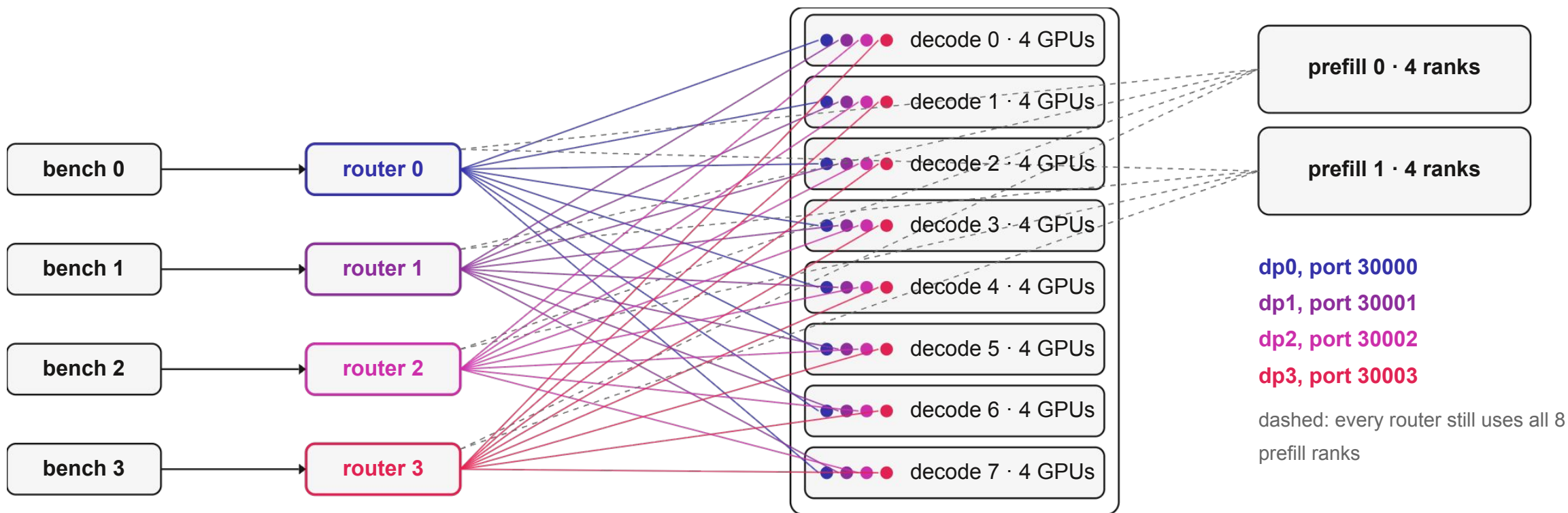
Failed requests, EP8, per 131k

~~240~~ 2

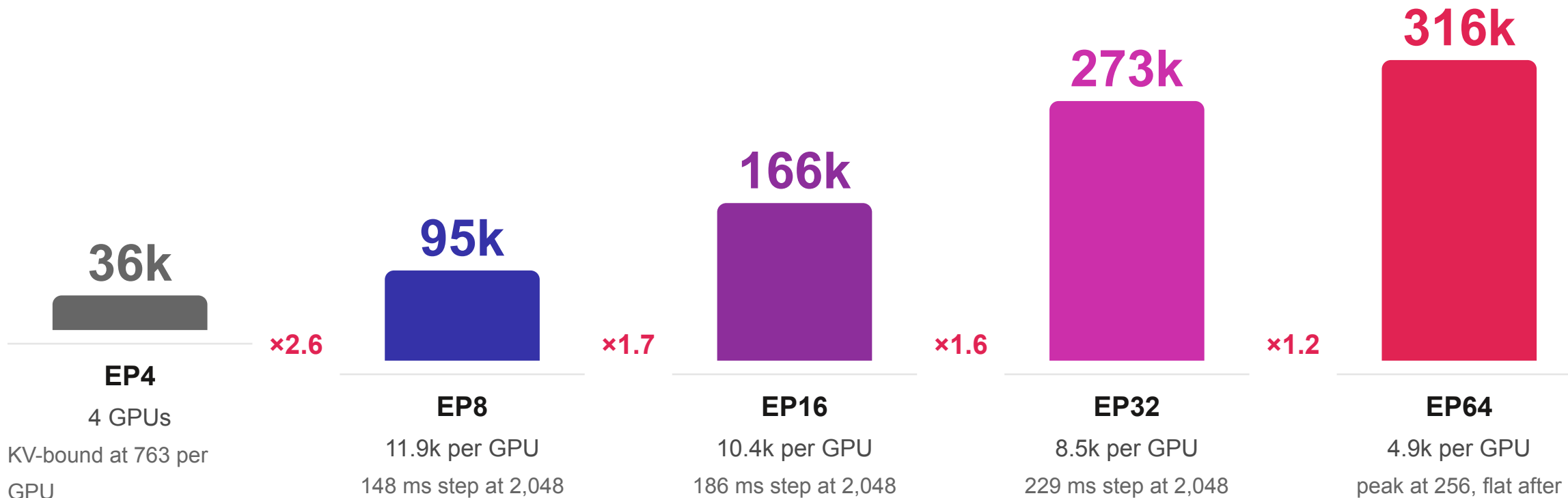
P99 TTFT, same point

~~102 s~~ 22 s

Lift request cap on sglang-router

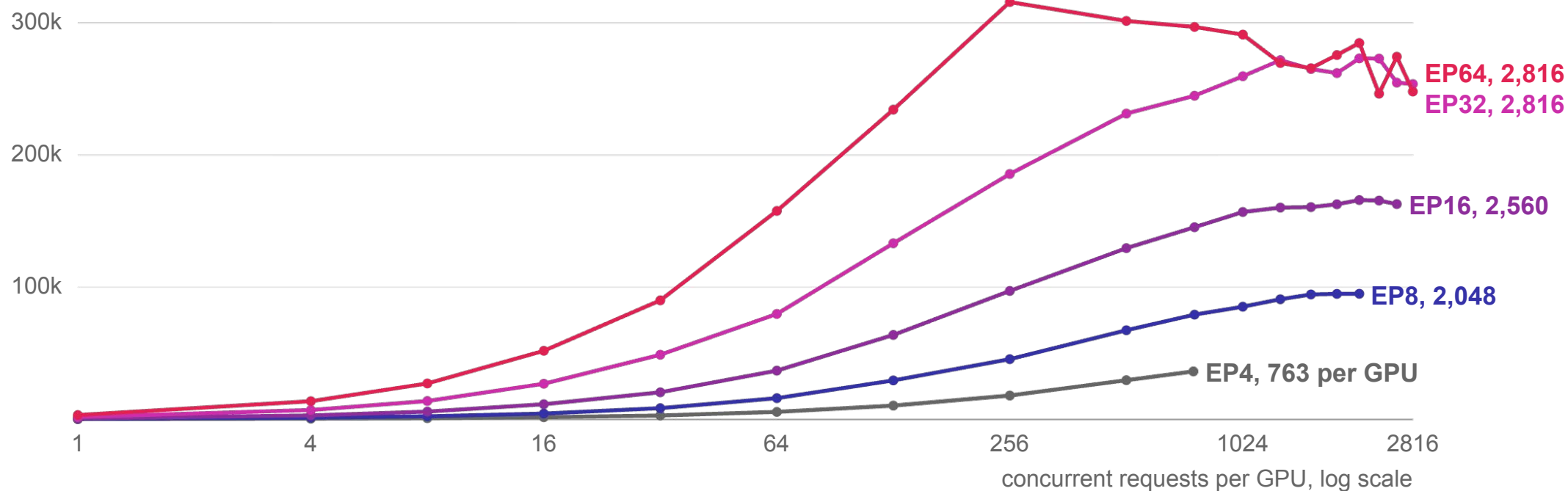


Peak throughput for each EP size

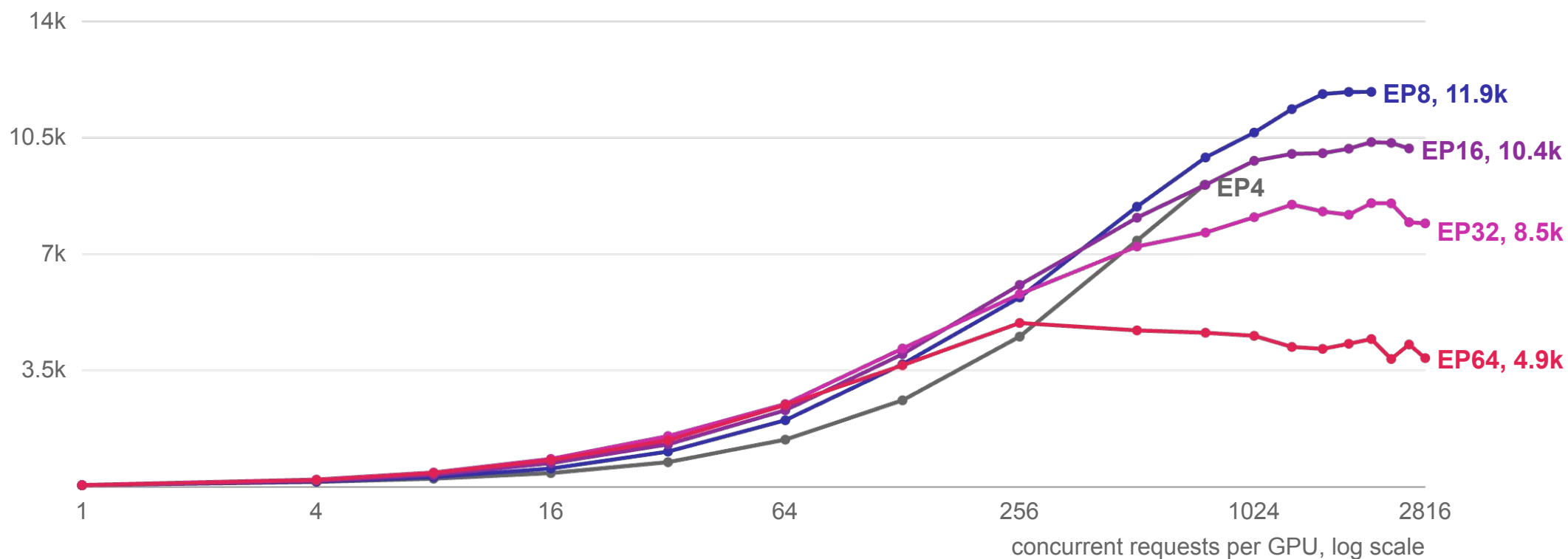


THROUGHPUT

Total Throughput for each EP size

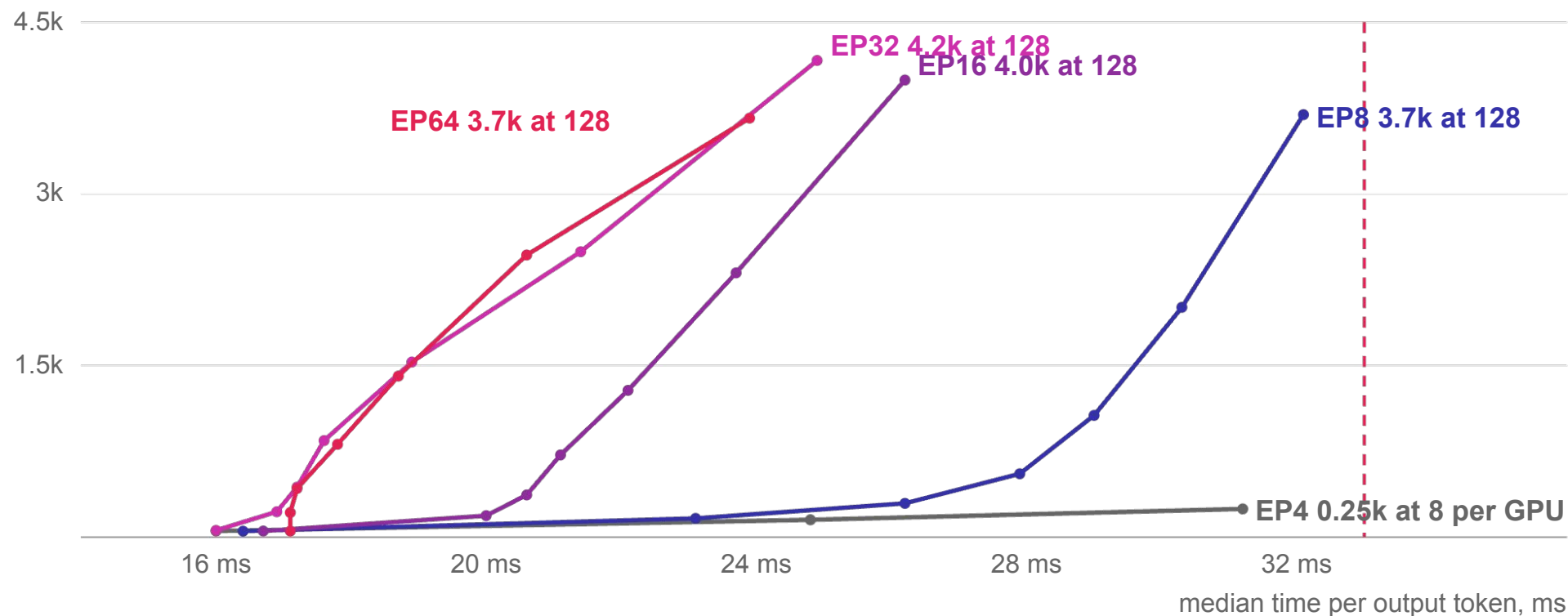


Per GPU throughput for each EP size



LATENCY BUDGET

Per GPU throughput in low latency(P99 ITL $\leq$ 33ms)



RECOMMENDATION

What to run on the rack today

EFFICIENCY

EP8 instances

11.9k

tokens/s per GPU, the best of any size

Independent all-to-all groups of 2 trays,
the lowest step cost per request.

LATENCY TARGET

EP32

4.2k

tokens/s per GPU under a 33 ms budget

The most per GPU within budget, at
128 requests per GPU.

NOT YET

EP64

+16%

over EP32, for twice the GPUs

4.9k tokens/s per GPU, the lowest of
any size.

All three numbers are a snapshot: adding speculative decoding and optimizing DeepEP performance will significantly change these results.