千问C端事业群
QWEN APPLICATIONS BUSINESS GROUP

ECHO: Elastic Speculative Decoding for High-Concurrency LLM Serving

ICML 2026 | Alibaba Qwen Booth

Xinyi Hu | Alibaba Qwen Applications Business Group

AGENDA

ECHO turns speculative decoding into a serving-time compute scheduling problem.

01

Why speculation fails under high concurrency

The verification-is-free assumption breaks in compute-bound serving.

02

ECHO: elastic compute-budget scheduling

Sparse confidence gates recover budget; a batch-level scheduler reallocates it.

03

Throughput and speedup results

Latency and throughput gains hold from single request to high concurrency.

04

Next steps

Native-MTP ECHO and context-aware Sweet Spot calibration.

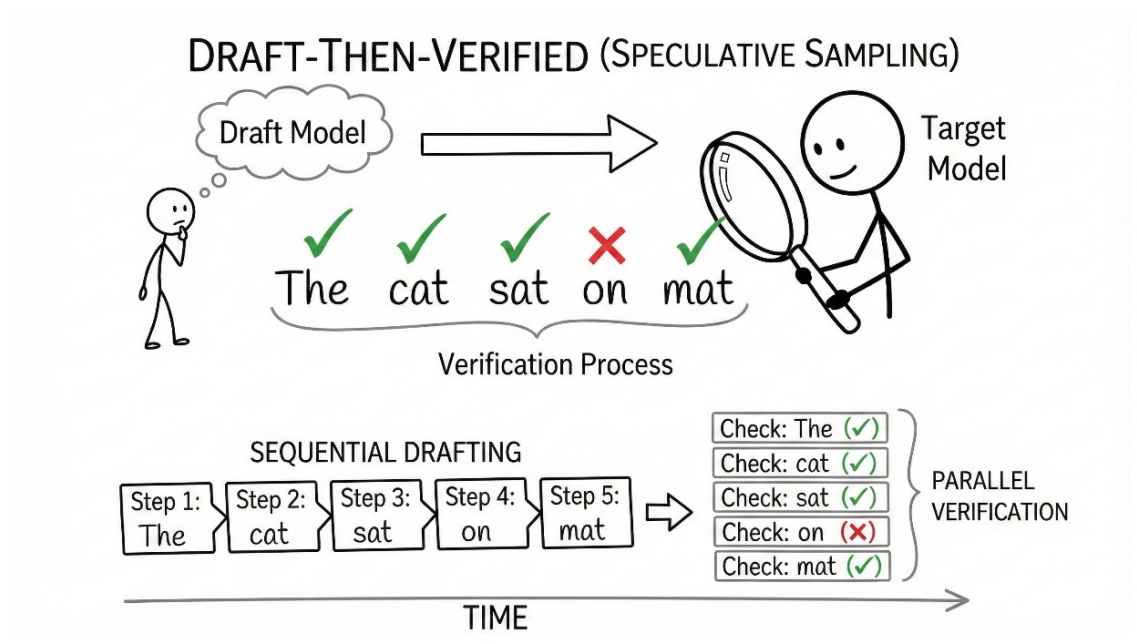
01

The Challenge in High-Concurrency Serving

Speculative decoding needs to spend draft compute where it actually converts into accepted tokens.

BACKGROUND

Speculative decoding works when verification is nearly free.



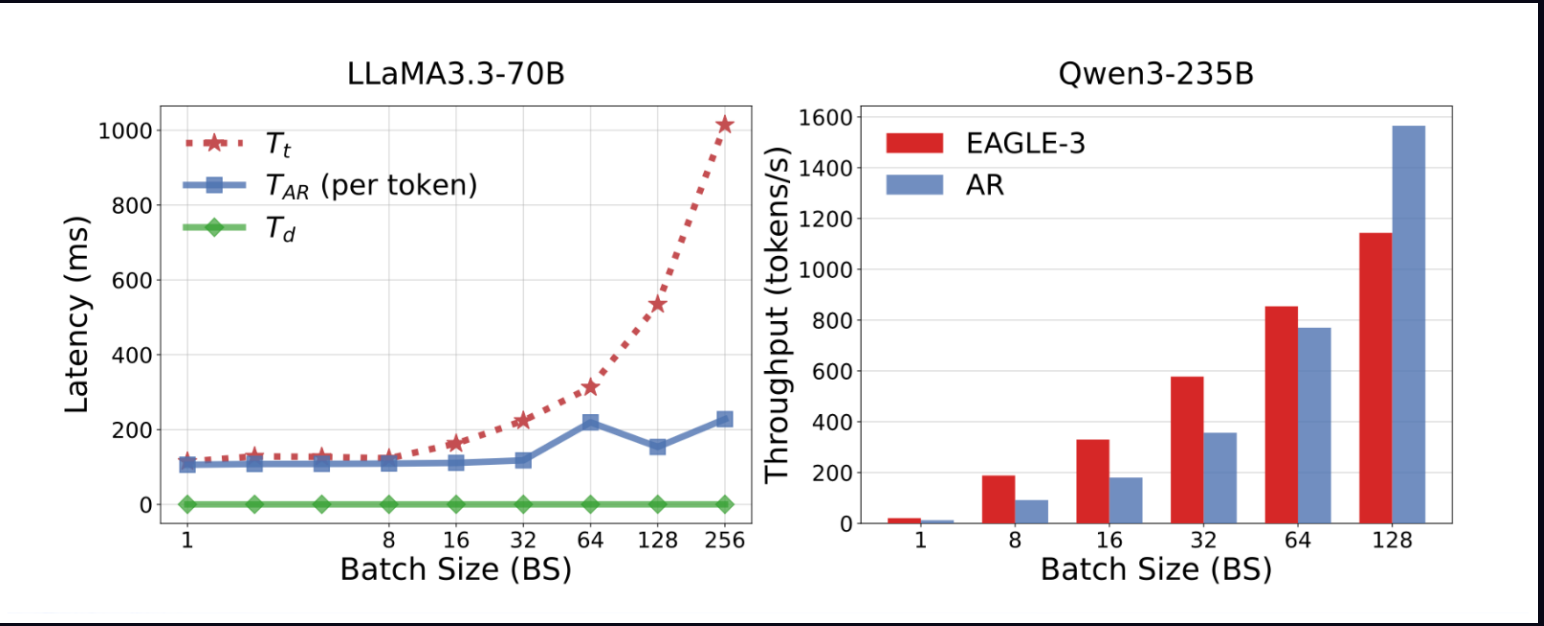
A draft model proposes multiple tokens, and the target model verifies them in one pass.

The core assumption: accepted drafts reduce decode steps without adding proportional verification cost.

In low-pressure regimes, the target pass is amortized across several draft tokens. ECHO starts from the question: what changes when serving is already compute-bound?

FAILURE MODE

Under compute-bound serving, extra draft tokens become real cost.



When batch size rises, each additional draft token consumes real compute.

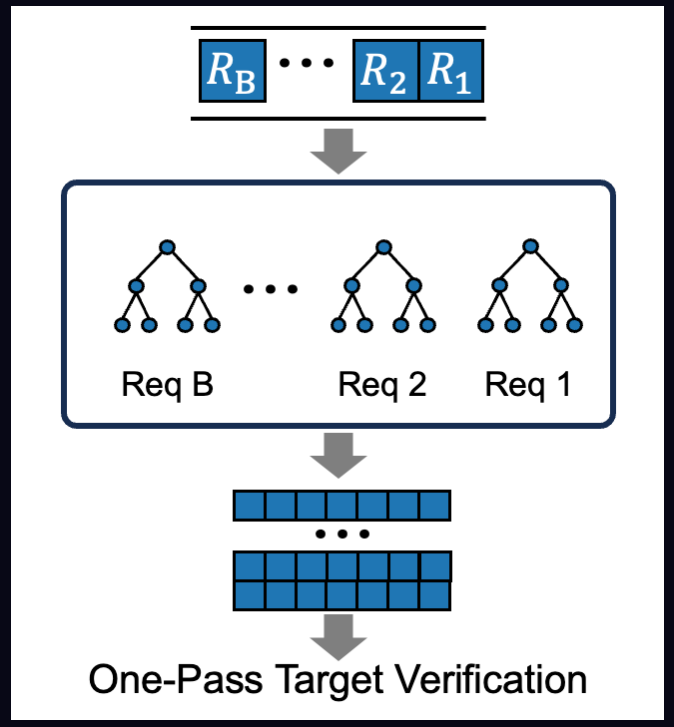
If acceptance does not keep up, speculation can reduce throughput instead of improving it.

compute-bound

Draft length must be budgeted, not blindly expanded.

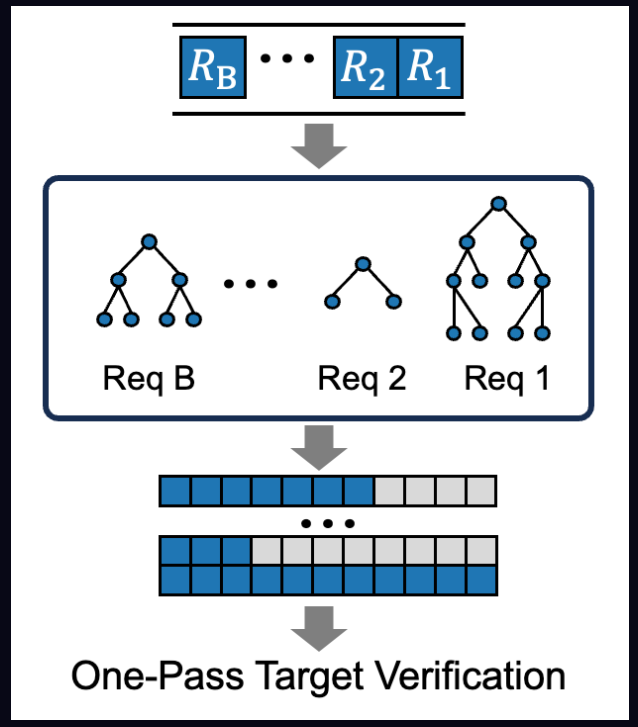
Per-request tree strategies miss the global compute tradeoff.

Static tree



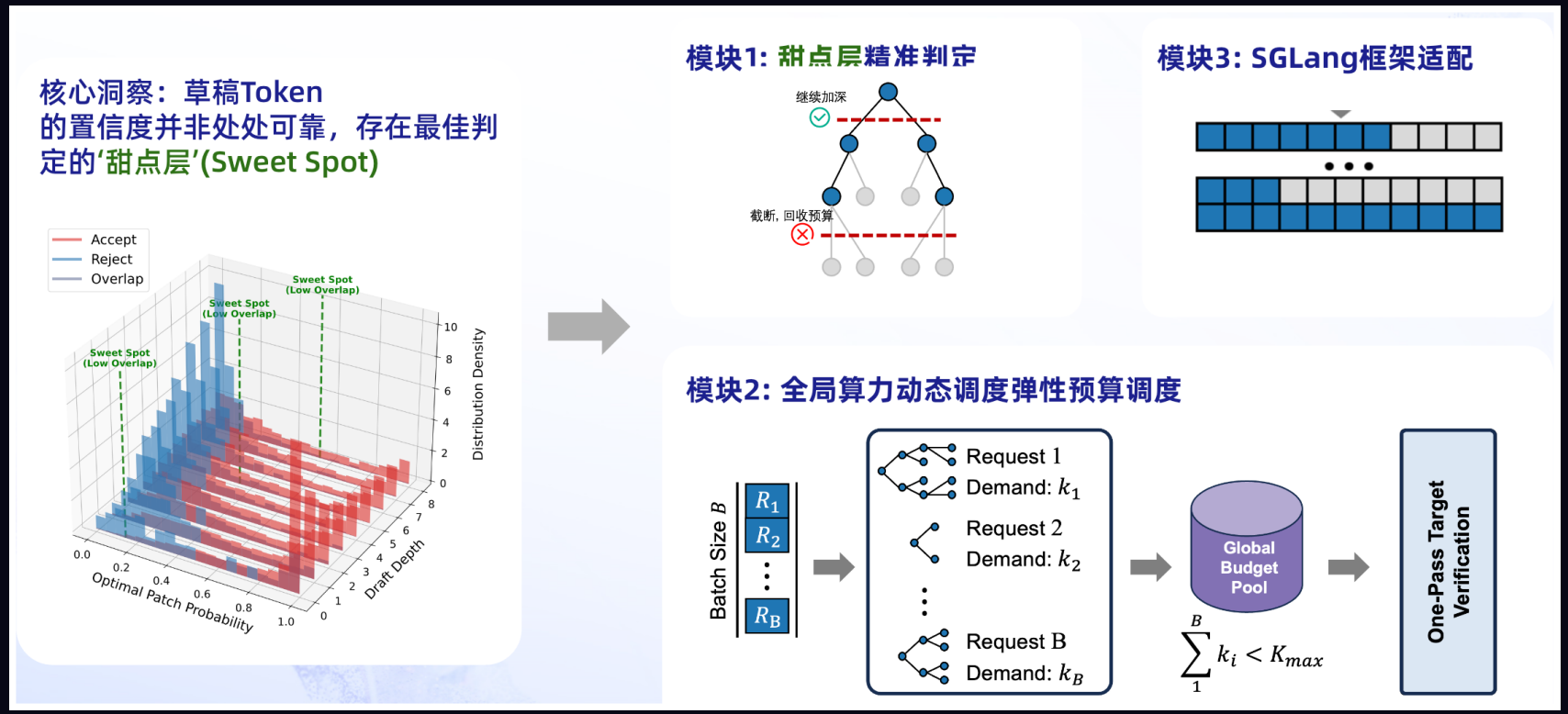
Framework-friendly, but wastes compute when acceptance signals are weak.

Dynamic tree



More adaptive per request, but does not balance compute across the whole batch.

ECHO reframes speculation as adaptive compute-budget scheduling.



Original solution layout from the source deck: Sweet Spot calibration, global elastic budget scheduling, and SGLang integration.

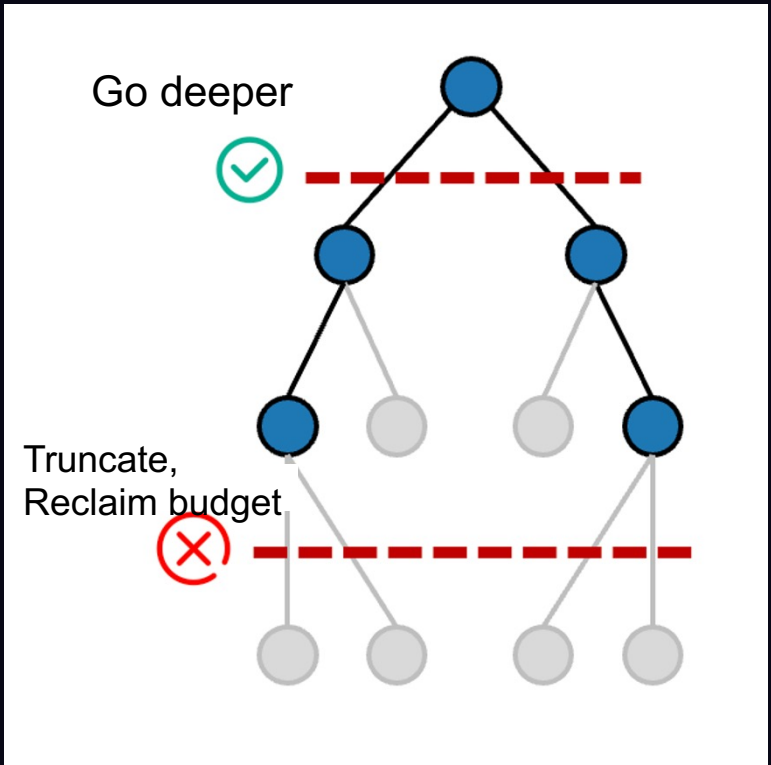
02

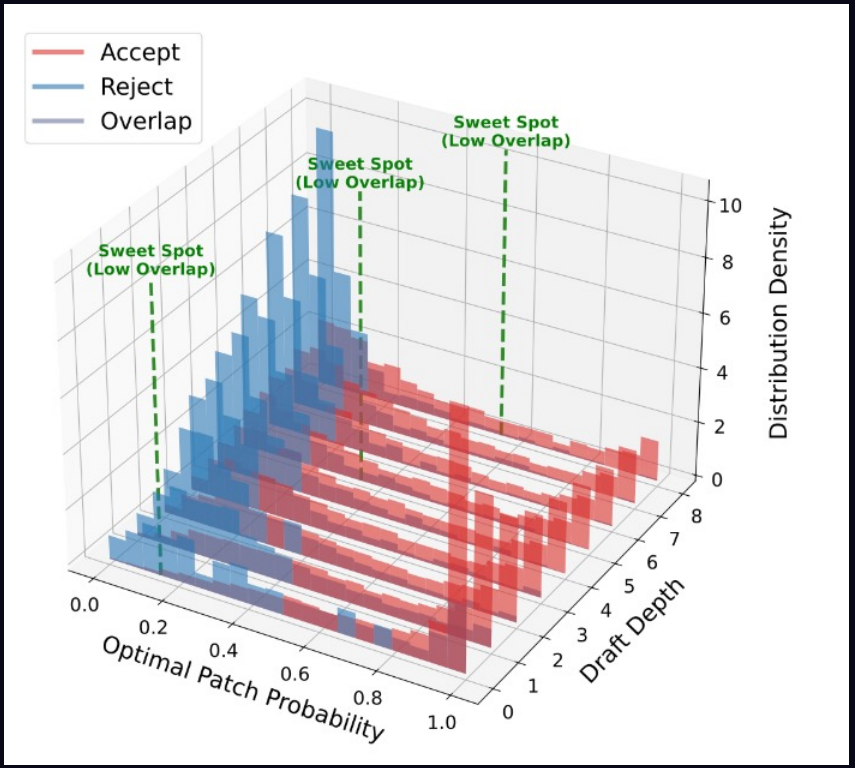
ECHO as Compute-Budget Scheduling

The mechanism has two levels: per-request sparse confidence gating and whole-batch elastic reallocation.

MECHANISM 1

Sparse confidence gating spends budget only at discriminative layers.



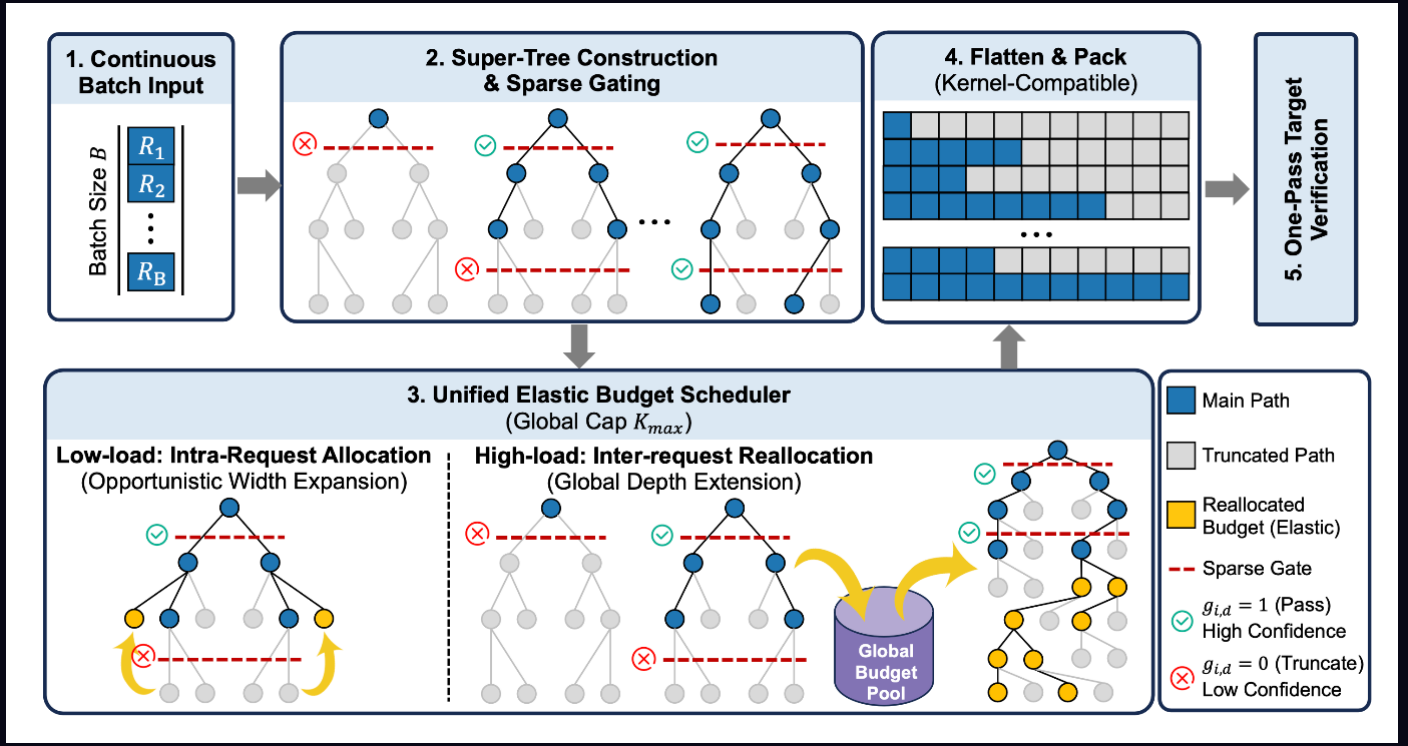


Warm-up calibration finds layers where confidence separates accepted and rejected drafts.

ECHO gates only at those layers to reduce error accumulation for each request.

MECHANISM 2

Elastic budget scheduling reallocates saved compute across the batch.



Priority 1: Go Deep to reduce verification rounds.

Priority 2: Go Wide to increase accepted length.

High load: inter-request scheduling.
Low load: intra-request allocation.

03

Throughput and Speedup Results

ECHO is evaluated across model scales, baselines, and serving loads.

BENCHMARK SETUP

Benchmarks cover model scale, baselines, and task diversity.

Models

Qwen3-8B / 32B / 235B
LLaMA3-8B / 70B
Vicuna-13B

Baselines

Static tree: EAGLE-3 (SOTA)
Dynamic tree: DDD, OPT-Tree

Datasets

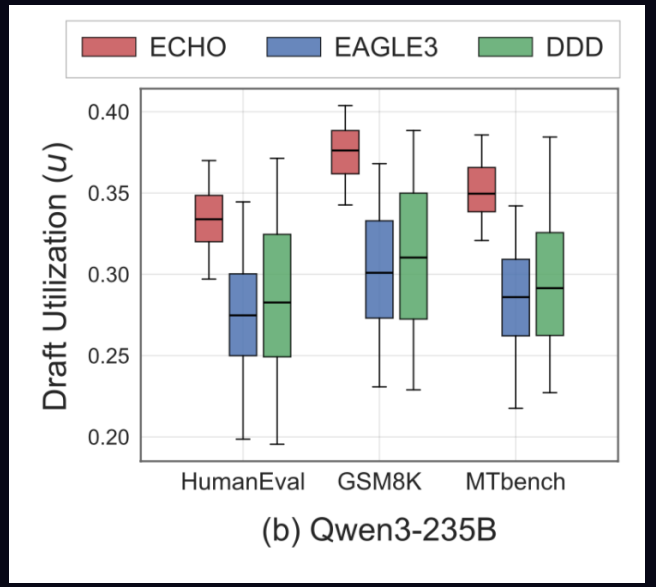
MT-Bench
GSM8K
HumanEval
Alpaca

The setup keeps both single-request latency and high-concurrency throughput in view, rather than optimizing one regime at the expense of the other.

LATENCY

At batch size 1, ECHO improves latency by raising draft-token utilization.

Models	Methods	HumanEval		GSM8K		CNN/DM		Alpaca		MT-Bench		Avg.
		MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	MAT	Speedup	
Vicuna-13B	Lookahead	1.73	1.69×	1.88	1.79×	1.48	1.44×	1.49	1.44×	1.67	1.63×	1.60×
	Sps	2.55	1.81×	1.99	1.75×	2.31	1.71×	2.01	1.74×	2.25	1.81×	1.76×
	Medusa	2.78	2.25×	2.63	2.12×	2.09	1.65×	2.44	1.96×	2.58	2.08×	2.01×
	Hydra	3.87	2.75×	3.66	2.60×	2.82	1.95×	3.51	2.48×	3.64	2.53×	2.46×
	OPT-Tree	8.21	3.85×	6.77	3.25×	6.91	2.75×	6.56	3.20×	6.95	3.30×	3.27×
	DDD	8.95	4.95×	6.21	3.92×	6.02	3.45×	5.98	3.59×	6.05	3.95×	3.97×
	EAGLE3	8.49	4.81×	6.82	3.85×	6.41	3.38×	6.49	3.65×	6.83	3.89×	3.92×
	ECHO	9.35	**5.25**×	6.53	**4.08**×	6.34	**3.53**×	6.24	**3.74**×	6.33	**4.12**×	**4.14**×
LLaMA3.1-8B	DDD	6.95	4.18×	6.02	4.07×	5.08	3.10×	6.53	4.08×	5.98	3.70×	3.83×
	EAGLE3	7.18	4.02×	6.50	3.94×	5.46	3.02×	7.06	4.01×	6.43	3.63×	3.72×
	ECHO	7.22	**4.30**×	6.28	**4.10**×	5.24	**3.26**×	6.81	**4.19**×	6.23	**3.86**×	**3.94**×
LLaMA3.3-70B	DDD	6.82	5.13×	6.15	4.63×	4.92	3.52×	6.68	4.84×	5.70	4.02×	4.43×
	EAGLE3	7.12	4.98×	6.53	4.72×	5.19	3.60×	6.95	4.75×	5.92	4.09×	4.43×
	ECHO	7.07	**5.35**×	6.41	**5.08**×	5.10	**3.79**×	6.84	**4.94**×	5.94	**4.32**×	**4.70**×
Qwen3-8B	DDD	3.65	2.54×	3.72	2.32×	3.08	2.13×	3.22	2.06×	3.48	2.35×	2.28×
	EAGLE3	3.91	2.37×	3.94	2.35×	3.28	1.98×	3.46	2.09×	3.71	2.20×	2.20×
	ECHO	3.82	**2.74**×	3.88	**2.68**×	3.20	**2.37**×	3.36	**2.51**×	3.63	**2.57**×	**2.57**×
Qwen3-32B	DDD	2.82	2.18×	3.15	2.38×	2.38	1.71×	2.68	2.11×	2.88	1.97×	2.07×
	EAGLE3	2.99	2.02×	3.32	2.41×	2.55	1.67×	2.83	1.97×	3.04	1.93×	2.00×
	ECHO	2.95	**2.37**×	3.29	**2.72**×	2.48	**1.93**×	2.74	**2.31**×	2.95	**2.25**×	**2.32**×
Qwen3-235B	DDD	2.20	1.88×	2.48	1.68×	2.05	1.49×	2.18	1.78×	2.28	1.99×	1.77×
	EAGLE3	2.41	1.82×	2.73	1.71×	2.02	1.35×	2.35	1.72×	2.54	1.83×	1.69×
	ECHO	2.32	**2.23**×	2.59	**1.92**×	2.08	**1.63**×	2.24	**2.07**×	2.37	**2.23**×	**2.02**×



5.35x
maximum observed speedup

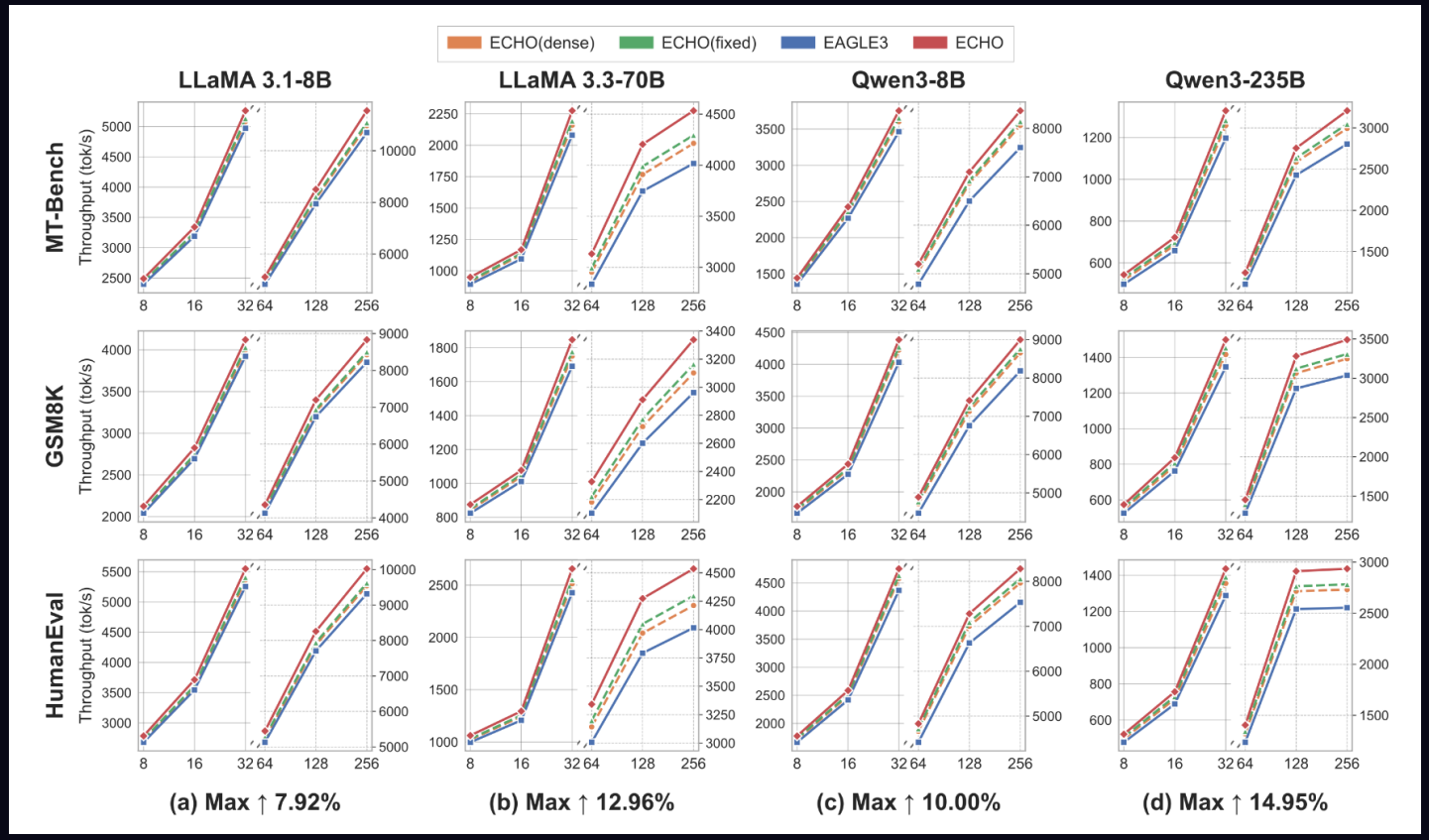
2.02x
on Qwen3-235B

+19.5%
vs. EAGLE-3

ECHO improves the utilization of drafted tokens: accepted length divided by speculative decoding depth.

CONCURRENCY

Under concurrency, ECHO keeps throughput gains across model sizes.



The gain is not a single-model artifact.

Across benchmark families and model sizes, the elastic budget policy keeps the accepted-token path productive under concurrent load.

SERVING TABLE

Qwen3-8B serving averages about 19% gain and peaks at 36%.

bs	steps	topk	num_draft_tokens	judge_depth	threshold	remaining_num_draft_tokens	throughput	eagle3 throughput	gains
4	5	8	32	3	0.3	20	521.28	457.43	14%
8	5	8	32	3	0.3	20	676.96	517.37	31%
16	5	8	32	3	0.3	20	749.22	549.91	36%
32	3	3	6	2	0.9	4	1909.19	1799.37	6%
64	3	3	6	2	0.9	4	2091.53	1900.13	10%

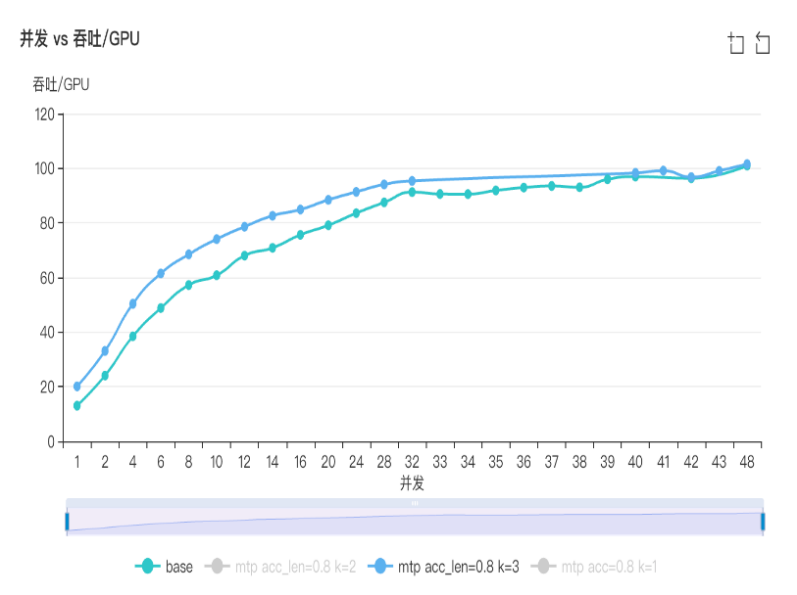
19%
average throughput gain

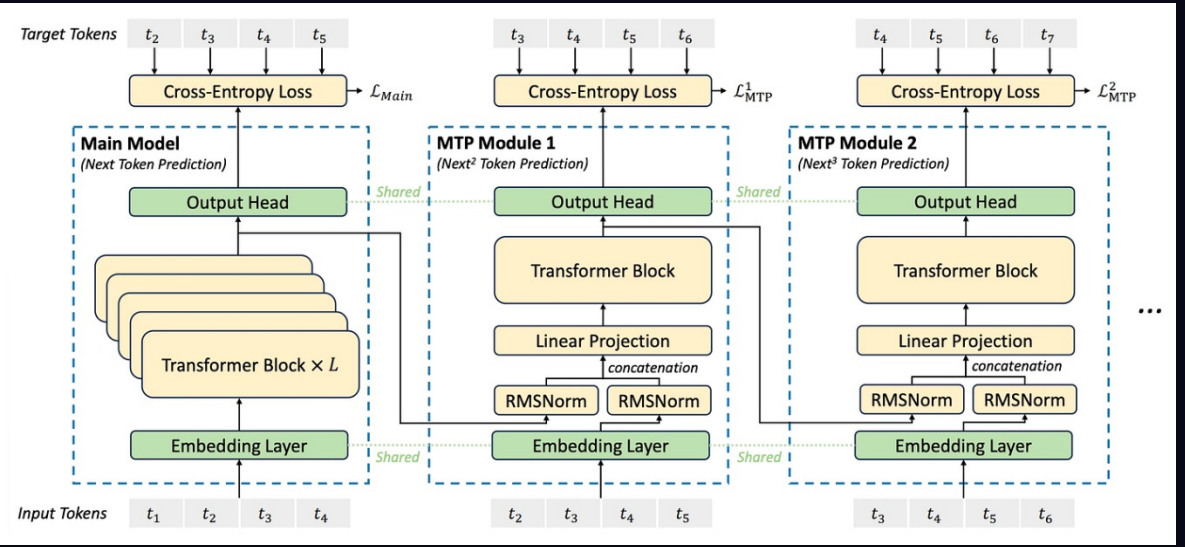
36%
maximum throughput gain

Configuration: Qwen3-8B, input length 100, output length 500.

QWEN3.5 ADAPTATION

Native-MTP deployment needs only light adaptation.





The MTP pipeline stays close to the EAGLE-style path, so ECHO can be inserted with minor scheduling changes.

As concurrency rises, native speculative execution can still lose efficiency; ECHO targets that gap.

04

Next Steps

Move from validated serving gains to native-MTP deployment and smarter calibration.

ROADMAP

Next work moves ECHO into native MTP and context-aware calibration.

01

Native-MTP ECHO

Integrate elastic speculative decoding with native MTP execution paths.

02

Context-aware Sweet Spot

Calibrate discriminative layers dynamically based on prompt and batch context.

03

Serving policy tuning

Jointly optimize latency, accepted length, and throughput under changing load.

THANK YOU

Questions and discussion