A Colab-Scale Study of Speculative Decoding: Practical Speed–Quality Frontiers Across Two Deployment Stacks on $1 \times$ A100

Stelios Zacharioudakis

March 2026

Abstract

Speculative decoding is widely discussed as a path to faster large language model (LLM) inference while preserving output quality. We report a Colab-scale study on a single NVIDIA A100 80GB GPU, comparing two practical deployment stacks: (i) Transformers-assisted decoding with a target–draft pair, and (ii) vLLM speculative decoding with an EAGLE-3 speculator. We measure latency, throughput, speedup versus baseline decoding, and proxy quality metrics under fixed per-study prompt sets and seeded protocols.

In Study 1 (Transformers-assisted, Qwen2.5 target), all tested configurations are slower than baseline (best QA speedup $0.355\times$), with only small QA proxy-F1 deltas versus baseline (ΔF1 from -0.0001 to -0.0005 relative to baseline F1 $= 0.0591$). In Study 2 (vLLM + EAGLE-3, Qwen3-8B target), speculative decoding shows consistent gains, with best overall speedup $1.387\times$ and category speedups up to $1.460\times$ on math-reasoning prompts.

Because the two studies differ in model family, prompt set, and decoding regime, the results should be interpreted as comparative evidence across realistic stacks rather than as a single controlled causal backend ablation. For this Colab/A100 setting, the strongest observed acceleration is in the $1.3\times$–$1.46\times$ range within the vLLM+EAGLE-3 study.

1 Introduction

Speculative decoding reduces autoregressive decoding latency by allowing a smaller or auxiliary model to propose tokens that the main model can verify in blocks. Foundational work reports substantial acceleration under favorable conditions [5, 2]. However, practical deployment often diverges from idealized settings due to runtime-specific costs, tokenizer constraints, and memory behavior.

This paper asks a practical question: *what speed–quality gains are reproducibly achievable in a Colab-scale environment on $1 \times$ A100?* We execute two studies with a shared reporting protocol:

- **Study 1:** Transformers-assisted speculative decoding with Qwen2.5 target/draft variants.
- **Study 2:** vLLM speculative decoding with Qwen3-8B target and RedHatAI EAGLE-3 speculator.

The main contribution is not a new algorithmic variant, but a systematic engineering evaluation that highlights where speedups materialize in practice and where they fail.

2 Related Work

Speculative decoding was formalized as a lossless acceleration strategy in [5], with subsequent sampling and systems variants [2, 9]. EAGLE-family methods shift speculation to stronger internal representations and dynamic trees [7, 8, 6]. Medusa provides a related multi-head acceleration alternative [1]. A recent survey synthesizes this rapidly expanding design space [12].

On the implementation side, practical behavior is governed by serving/runtime stack details, including tokenizer compatibility, batching strategy, and kernel/runtime scheduling [4, 3, 11, 10].

3 Experimental Setup

3.1 Hardware and Runtime

- GPU: NVIDIA A100-SXM4-80GB (Google Colab runtime)
- Single-GPU inference (no tensor parallelism across devices)
- Seeded benchmarking with explicit warmup excluded from metrics

3.2 Study 1 (Transformers-Assisted)

- Target: `Qwen/Qwen2.5-7B-Instruct`
- Drafts: `Qwen/Qwen2.5-0.5B-Instruct` (close-match), `Qwen/Qwen2.5-0.5B` (mismatch)
- Prompt set: 24 SQuAD-derived prompts (short/long/QA splits)
- Generation length: 128 new tokens
- Block-size ablation: {8,16} plus mismatch at 16
- Decoding settings: `do_sample=False`, `temperature`=1.0

3.3 Study 2 (vLLM + EAGLE-3)

- Target: `Qwen/Qwen3-8B`
- Speculator: `RedHatAI/Qwen3-8B-speculator.eagle3`
- Prompt set: 24 prompts (12 summarization + 12 math reasoning) from the Red Hat AI `speculator_benchmarks` dataset
- Speculative token ablation: {3,5}
- Decoding settings: `temperature`=0.6, `top_p`=0.95, `top_k`=20, seed=0

3.4 Reproducibility Profile

- Runtime: Google Colab Python 3.12 on $1 \times$ A100-80GB.
- Library versions used in the notebook: PyTorch 2.8.0+cu128, Transformers 4.57.6, vLLM 0.11.0.
- Study 1 assisted-decoding parameters: max_new_tokens=128, num_assistant_tokens in {8,16}, assistant_confidence_threshold=0.4.

Table 1: Study 1 (Transformers-assisted) results on QA split. Baseline QA latency: 4.052 s, baseline global throughput: 31.589 tok/s.

Config	Speedup$_{QA}$	Latency$_{QA}$ (s)	Quality F1	Acceptance Proxy
close_match_b8	0.310	13.053	0.0590	0.280
close_match_b16	0.312	12.971	0.0590	0.164
mismatch_b16	**0.355**	**11.429**	0.0586	0.169

Table 2: Study 2 (vLLM + EAGLE-3) aggregate results.

Config	Num Spec Tokens	Speedup	Global TPS	Quality F1
eagle3_n3	3	**1.387**	**110.926**	**0.2880**
eagle3_n5	5	1.325	105.903	0.2830

- Study 2 vLLM parameters: max_new_tokens=128; num_speculative_tokens in {3,5}; max_model_len=4096; gpu_memory_utilization=0.55.
- Seed control: Python/NumPy/PyTorch seed=0; vLLM SamplingParams seed=0.

3.5 Metrics

We report:

- **Latency** (mean and median)
- **Throughput** (global tokens/sec)
- **Speedup** vs baseline:

$$\text{Speedup} = \frac{T_{\text{baseline}}}{T_{\text{speculative}}} \tag{1}$$

- **Quality proxies**: normalized-token F1-like score and exact-match-vs-baseline checks (proxy-only)
- **Acceptance proxy**: token-prefix agreement rate between baseline/speculative outputs

4 Results

4.1 Study 1: Assisted Decoding Did Not Accelerate

Table 1 shows that all tested assisted-decoding configurations were slower than baseline (speedup < 1), with best QA speedup 0.355×.

Quality remained close to baseline: baseline QA F1 is 0.0591, and ΔF1 vs baseline is -0.0001 (close_match_b8), -0.0001 (close_match_b16), and -0.0005 (mismatch_b16). This suggests the regression is primarily systems/runtime overhead rather than output collapse on this QA proxy.

4.2 Study 2: vLLM + EAGLE-3 Produced Practical Speedups

The vLLM+EAGLE-3 stack produced materially better latency/throughput behavior (Table 2). Best speedup reached **1.387**× (`eagle3_n3`).

Table 3: Latency by category for best Study 2 config (`eagle3_n3`).

Category	Baseline Latency (s)	Speculative Latency (s)	Speedup
math_reasoning	1.575	1.079	**1.460**
summarization	1.627	1.229	1.323

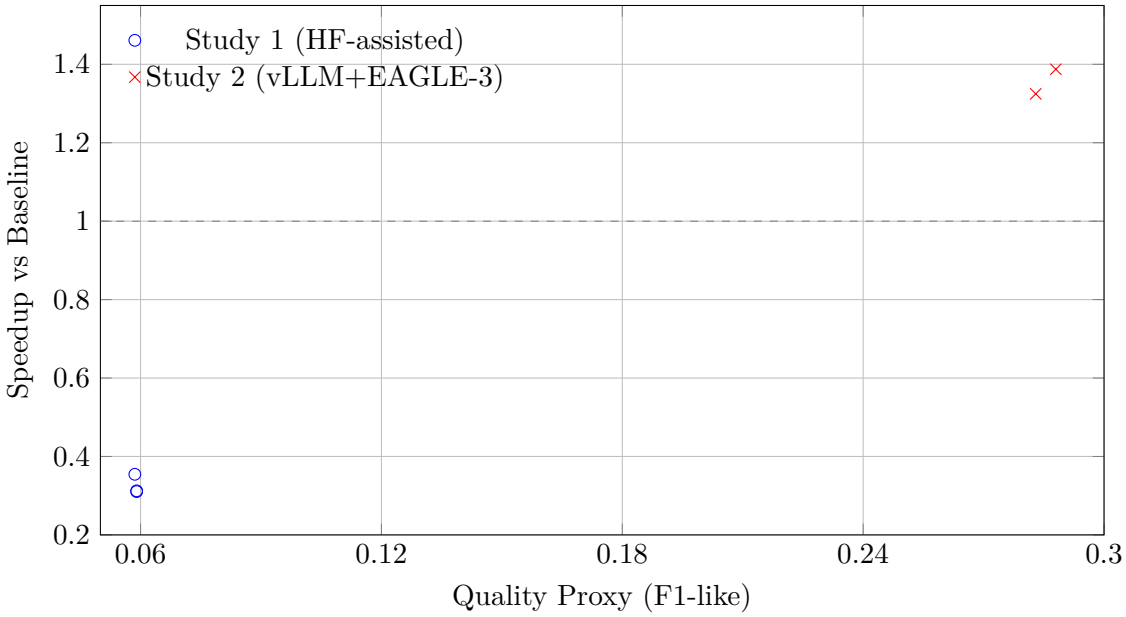

Figure 1: Speed–quality frontier across the two studies. Since models/prompts/regimes differ across studies, this figure is comparative (not a controlled causal backend ablation).

Per-category latency analysis for the best configuration is shown in Table 3.

Study 2 quality should be interpreted as proxy-only: the reported F1 is reference-relative on a different task mix than Study 1, and exact match vs baseline is 0.0 under seeded stochastic decoding.

4.3 Internal Stability Slice (Study 2)

Although we did not run full multi-seed repeats, Study 2 shows consistent acceleration across two independent slices: both tested speculation budgets exceed 1× (1.387 and 1.325), and both prompt categories exceed 1× (math 1.460, summarization 1.323).

5 Discussion

Why Study 1 underperformed. Our Study 1 setup required universal tokenizer handling and incurred significant runtime overhead in assisted generation, overwhelming theoretical savings from draft proposals in this environment.

Why Study 2 improved. vLLM with EAGLE-3 delivered substantially better kernel/runtime behavior and scheduling in this stack, yielding practical gains above 1.3×.

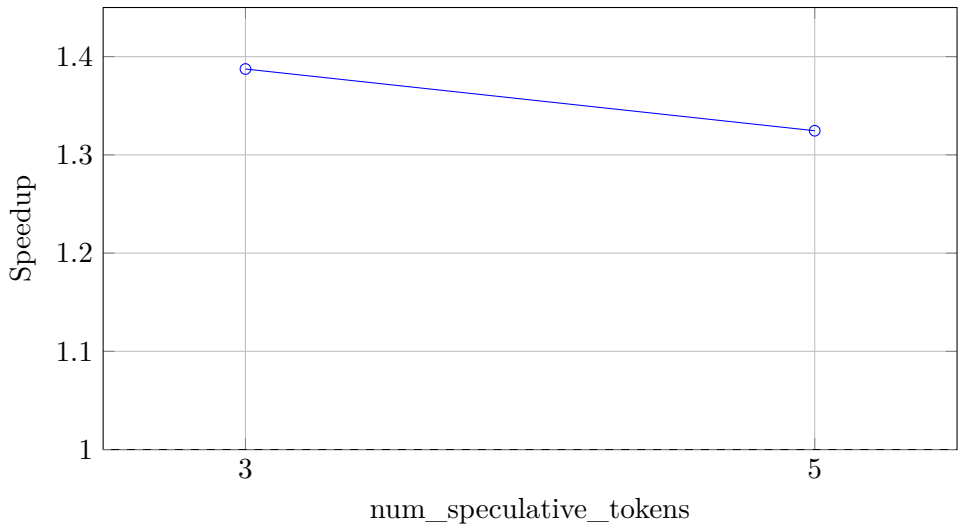

Figure 2: Study 2 ablation over speculative token budget. In this run, 3 tokens outperformed 5.

Why not 2 ×–3 × yet. Even with optimized serving, observed gains remain bounded by prompt-length distribution, speculation budget, and acceptance dynamics. Under the tested conditions, we observed up to 1.460× category speedup and 1.387× aggregate speedup, not 2×–3×.

6 Limitations and Threats to Validity

- Small evaluation set (24 prompts) chosen for rapid Colab iteration.
- Cross-study confounds exist: studies differ in target model family, prompt set, and decoding regime.
- Quality metrics are proxy-based and task-heterogeneous (SQuAD-style QA proxy in Study 1; reference-relative prompt mix in Study 2).
- No full multi-seed repeated-run error bars were computed; stability evidence is limited to ablation/category consistency within Study 2.
- Single hardware/runtime profile (1 × A100, Colab).
- Results can shift with newer runtime/library versions and different batching/prompt distributions.

7 Conclusion

This work provides a practical speculative-decoding benchmark in a Colab-scale environment across two realistic deployment stacks. Study 1 (Transformers-assisted) regressed in latency despite near-baseline QA proxy quality, while Study 2 (vLLM+EAGLE-3) reached 1.387× aggregate speedup and up to 1.460× on math prompts. The claim we support is comparative and stack-specific: in this setup, the strongest measured gains are in the ∼1.3–1.46× range, with 2×–3× remaining a target for broader systems tuning and larger-scale serving conditions.

References

[1] Tianle Cai, Yuhong Li, Zhengyang Geng, et al. Medusa: Simple llm inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[2] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, et al. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

[3] Hugging Face. Universal assisted generation. Hugging Face Blog, 2024.

[4] Hugging Face. Hugging face transformers documentation: Assisted generation. Hugging Face Docs, 2026.

[5] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[6] Yuhui Li et al. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.

[7] Yuhui Li, Feng Wei, Jiajun Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

[8] Yuhui Li, Hongyang Zhang, et al. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024.

[9] Jonathan Mamou, Miguel Del Rio, Brian Stephenson, et al. Dynamic speculation lookahead accelerates speculative decoding of large language models. *arXiv preprint arXiv:2405.04304*, 2024.

[10] Red Hat AI. Redhatai qwen3-8b-speculator.eagle3 (model card). Hugging Face Model Card, 2026.

[11] vLLM Team. vllm documentation: Speculative decoding. vLLM Documentation, 2026.

[12] Heming Xia, Zhe Yang, et al. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of ACL 2024*, 2024.