

Efficient Language-Model Inference at the Edge

13 September 2026 | Technical review | Version 1.0
QDOX Research | QDOX-TR-02

Public-source technical review and proposed methods. No newly completed experiments.
Independent publication. Not peer reviewed by IEEE.

Abstract

Interactive machine intelligence places different demands on inference systems than offline batch processing. A useful design must consider response latency, memory capacity, energy, numerical behavior, and interference with other workloads. This review connects public work on the Roofline model, IO-aware attention, heterogeneous offloading, grouped-query attention, and speculative decoding. It develops a compact accounting framework for memory and execution time, then proposes a reproducible evaluation protocol. An explicitly hypothetical calculation illustrates why speculation can either accelerate or slow generation depending on acceptance and verification cost. The article contains no new device benchmark, trained model, robot experiment, or company implementation result. Its contribution is a synthesis of public methods and a testable evaluation plan.

Index Terms: Edge inference, speculative decoding, heterogeneous computing, memory bandwidth, verification, performance evaluation.

I. Introduction

An interactive assistant and an offline document-processing service can use the same language model while requiring different systems. Aggregate throughput rewards batching; interaction also depends on the wait for the first response and the spacing between subsequent tokens. A system should therefore be evaluated against a defined use case rather than a single tokens-per-second figure.

Three questions organize this review. First, which memory hierarchy or execution resource limits an inference workload? Second, when does drafting several candidate tokens justify additional computation? Third, what evidence is needed before an optimization can be considered suitable for a shared edge device? The proposed protocol treats these as measurable questions. It does not presume that a CPU/GPU split, a persistent kernel, or a longer draft is beneficial.

II. Related Work

Roofline relates attainable arithmetic throughput to compute capacity, memory bandwidth, and operational intensity. Its value is diagnostic: a proposed optimization should address the limiting resource rather than improve an unrelated peak metric. [1]

FlashAttention demonstrates an IO-aware approach to exact attention by tiling work to reduce transfers between accelerator memory and on-chip storage. This motivates accounting for where bytes move, rather than treating operation count as a complete cost model. It does not imply a universal end-to-end improvement across all decoding workloads or devices. [2]

FlexGen explores limited GPU-memory inference using GPU, CPU, and disk resources. Its emphasis on latency-insensitive batch processing is an important boundary: a throughput-oriented offloading result is not automatically suitable for interactive robotics.

Grouped-query attention changes another part of the problem by sharing key/value heads among query heads. Model architecture and execution scheduling therefore affect different costs. [3] [4]

Speculative decoding and speculative sampling use cheaper proposals followed by target-model verification. Their distribution-preserving forms require a correct acceptance and correction rule. They provide a foundation for conditional acceleration, rather than a guarantee that more speculative work is always faster. [5] [6]

III. Resource and Scheduling Model

A. Memory capacity and traffic

A useful memory budget distinguishes resident capacity from traffic during execution:

$$M_{\text{resident}} = M_W + M_{KV} + M_A + M_R. \quad (1)$$

The terms denote weights, key/value cache, temporary activations, and runtime storage. For a dense cache with equal-length sequences, element counting gives

$$M_{KV} = 2LbTH_{KV}d_h s. \quad (2)$$

Here, L is layer count, b batch size, T cached sequence length, H_{KV} key/value heads, d_h head dimension, and s bytes per stored element. The factor two accounts for keys and values. This estimate excludes padding, allocator overhead, quantization metadata, and architecture-specific compression. Sliding windows and variable-length batches require corresponding changes. Grouped-query attention affects H_{KV} ; a faster scheduler does not itself reduce that architectural count. [4]

Let F be arithmetic work and Q bytes transferred across a specified memory boundary during the same measurement window. Operational intensity is $I = F/Q$. A Roofline-style bound is

$$P_{\text{attainable}} \leq \min(P_{\text{compute}}, BI), \quad T \geq \max(F/P_{\text{compute}}, Q/B). \quad (3)$$

B must describe the relevant sustained bandwidth. Counting only weight reads while dividing by a total-system counter creates a scope mismatch. Dependencies, synchronization, and imperfect utilization can make execution slower than this bound. [1]

B. Heterogeneous execution

Shared addressing simplifies programming but does not eliminate coherence costs or establish safe concurrent access. CUDA documents different unified-memory behavior across devices and platforms. Atomicity depends on memory type, supported capabilities, and synchronization scope. A proposed CPU/GPU work queue must satisfy those conditions before its latency is meaningful. [7] [8]

For a proposed dependency graph, a basic lower bound is

$$T_{\text{step}} \geq \max(T_{\text{critical}}, F_C/P_C, F_G/P_G, Q_{\text{shared}}/B_{\text{shared}}). \quad (4)$$

This is an analytical scheduling bound, not a measured prediction. The subscripts distinguish CPU, GPU, and a shared memory bottleneck. Independent peak CPU and GPU bandwidths must not simply be added when both consume the same constrained channel. Additional transfers across a separate interconnect introduce another bound. A queue optimization can reduce coordination costs while leaving insufficient independent work unchanged.

IV. Speculation and Correctness

Let $p(x)$ be the target distribution and $q(x)$ the proposal distribution for the same prefix. A proposed token drawn from q can be accepted with probability

$$a(x) = \min \{1, p(x)/q(x)\}. \quad (5)$$

Following rejection, the correction distribution is proportional to $\max \{0, p(x) - q(x)\}$. Properly applying the complete algorithm preserves the target distribution. Greedy acceptance is a different contract: it compares token choices against a deterministic target path. Approximate arithmetic, altered sampling rules, and state-management errors must be evaluated separately. [6]

For k draft tokens and independent, identically distributed acceptance events with mean α , the expected emitted-token count is

$$E[N] = \sum_{i=0}^k \alpha^i = \frac{1 - \alpha^{k+1}}{1 - \alpha}, \quad \alpha \neq 1, \quad (6)$$

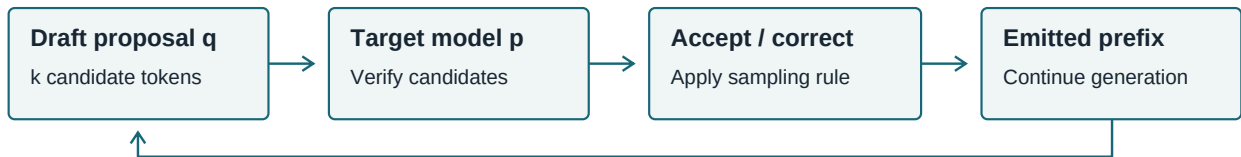
with limit $k + 1$ at $\alpha = 1$. This simplified model does not describe every prompt's correlated acceptance behavior. [5]

An accounting extension for planning experiments is

$$S_{\text{model}}(k) = \frac{E[N]T_1}{kT_D + T_V(k+1) + T_H}. \quad (7)$$

T_1 is baseline per-token time; T_D is draft cost; T_V is verification cost; and T_H covers acceptance and state handling. This serial-cost model assumes no overlap among those terms. For an overlapped implementation, replace the denominator with the measured round critical path. A useful draft length must improve the complete ratio, not acceptance alone.

A. Public method: propose, verify, accept or correct



B. Analytical accounting: one serial round

Drafting: $k T_D$	Verification: $T_V(k+1)$	State handling: T_H
-------------------	--------------------------	-----------------------

$$\text{Round cost} = k T_D + T_V(k+1) + T_H$$

Conceptual method and serial-cost model only. No measured timing or company implementation is depicted.

Fig. 1. Conceptual speculative-decoding workflow and serial round-cost model. The target verifies proposed tokens; acceptance and correction preserve the declared sampling contract. Stage widths are schematic, not measured durations. The diagram represents public methods, not a company implementation.

V. Analytical Illustration

The following values are assigned solely to illustrate equation (7). They are not measurements or predictions for a particular device. Choose $k = 4$, $T_D = 0.05T_1$, $T_V(5) = 1.4T_1$, and $T_H = 0.1T_1$. The modeled round costs $1.7T_1$.

TABLE I
Assigned values for an analytical illustration; not measurements

Assigned acceptance α	Expected tokens per round	Modeled speed ratio
0.2	1.2496	0.7351
0.6	2.3056	1.3562
0.9	4.0951	2.4089

At low acceptance, additional work outweighs token reuse. At higher acceptance, the same assumed costs yield a benefit. The illustration motivates measuring acceptance and verification latency together; it provides no evidence about actual workloads.

VI. Proposed Evaluation Protocol

This section proposes future work. It does not report an executed campaign.

1. **Define the comparison.** Record a public model revision and license, tokenizer, precision, prompt set, context lengths, output limits, sampling settings, and software versions. Keep baseline and candidate quality requirements identical.
2. **Separate timing windows.** Report model loading, prompt processing, time to first token, per-token latency, and complete request latency separately. Use the same window for work, byte, energy, and time ratios. Include median and tail latency alongside aggregate throughput.
3. **Vary one mechanism.** Compare ordinary decoding with speculation; compare specified scheduling policies using identical model inputs. Vary draft length and concurrent workload intensity independently before studying interactions. Retain unsuccessful configurations.
4. **Verify execution.** Check process exit status, numerical reference comparisons, worker participation, state reset, cancellation, and repeated execution. Validate distribution-preserving sampling with appropriate statistical tests rather than assuming token-by-token identity under one random seed.
5. **Instrument carefully.** Public CUDA tools provide memory, initialization, synchronization, and shared-memory hazard checks. A clean instrumented run covers the paths and error classes exercised; it does not establish a complete correctness or security proof. Timing collected under instrumentation must be distinguished from ordinary execution. [9]
6. **Test interaction constraints.** Measure inference alongside a separately defined periodic workload. Report deadline misses, resource use, thermal conditions, and recovery after cancellation. A language-model throughput result alone does not establish suitability for actuator control.

The proposed publication record should include public configuration files, raw measurements where consent and licensing permit, analysis scripts, environment information, and precise exclusions. A graph of latency against context length is more informative than an isolated best

score. A timeline of drafting, verification, synchronization, and other device work can help explain a result, provided profiler overhead is characterized. [10]

VII. Discussion and Limitations

These models expose tradeoffs without selecting a universal implementation. Memory-saving techniques can add computation; a batch that improves throughput can worsen response delay; a heterogeneous schedule can add coordination overhead. A capacity calculation does not measure traffic, a theoretical bound does not predict tail latency, and numerical equivalence does not prove physical-system safety.

The review is selective rather than a systematic survey. It uses public foundational papers and vendor documentation to construct a reproducible research plan. No result here establishes proprietary implementation progress, device ownership, model quality, secure serving, robotic autonomy, or deployment readiness. Those require separate evidence from an explicitly authorized experiment.

VIII. Conclusion

Efficient edge inference should be evaluated as a complete interaction: memory, computation, scheduling, numerical behavior, and the demands of other work on the device. Public research supplies useful mechanisms and analytical tools. A credible next step is a controlled, openly documented comparison that measures when each mechanism helps and when it fails.

References

- [1] S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Floating-Point Programs and Multicore Architectures," University of California, Berkeley, Tech. Rep. UCB/EECS-2008-134, 2008. [Public report](#). Related journal article: *Communications of the ACM*, vol. 52, no. 4, pp. 65-76, 2009, DOI: 10.1145/1498765.1498785.
- [2] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness," *Advances in Neural Information Processing Systems*, vol. 35, 2022. [Public paper](#).
- [3] Y. Sheng et al., "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU," in *Proc. 40th International Conference on Machine Learning*, PMLR, vol. 202, pp. 31094-31116, 2023. [Public paper](#).
- [4] J. Ainslie et al., "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints," in *Proc. EMNLP*, pp. 4895-4901, 2023, DOI: 10.18653/v1/2023.emnlp-main.298. [Public paper](#).
- [5] Y. Leviathan, M. Kalman, and Y. Matias, "Fast Inference from Transformers via Speculative Decoding," in *Proc. 40th International Conference on Machine Learning*, PMLR, vol. 202, pp. 19274-19286, 2023. [Public paper](#).
- [6] C. Chen et al., "Accelerating Large Language Model Decoding with Speculative Sampling," arXiv:2302.01318, 2023. [Public preprint](#).
- [7] NVIDIA, "Unified and System Memory," *CUDA Programming Guide*, accessed Sep. 2026. [Official documentation](#).
- [8] NVIDIA, "CUDA C++ Memory Model," *CUDA Programming Guide*, accessed Sep. 2026. [Official documentation](#).
- [9] NVIDIA, *Compute Sanitizer*, accessed Sep. 2026. [Official documentation](#).
- [10] NVIDIA, *Nsight Systems User Guide*, accessed Sep. 2026. [Official documentation](#).

Preparation Note

OpenAI Codex assisted with public-source review, drafting, equations and original vector diagrams. This document reports no newly executed hardware or robot experiment. Its structure borrows conventions from IEEE author guidance; no IEEE submission, acceptance, endorsement or peer review is represented.