Speculative Decoding

Accelerating LLM Inference with Draft Models

February 2026

Table of Contents

1. Introduction

Autoregressive decoding is the fundamental bottleneck of LLM inference. Each token requires a full forward pass through the model, and these passes are strictly sequential—token N+1 cannot be generated until token N is known. For large models distributed across multiple GPUs, each step also incurs inter-device communication overhead. Speculative decoding is a family of techniques that break this sequential bottleneck by generating multiple candidate tokens cheaply and verifying them in parallel against the target model.

The core insight is simple: a small, fast draft model can predict what the large target model would say with surprisingly high accuracy for routine text. When the draft is correct, multiple tokens are accepted in a single target-model forward pass. When it is wrong, the system falls back gracefully to standard decoding with no quality loss. The result is a 2–3x reduction in the number of target-model forward passes, translating directly into lower latency and fewer rounds of inter-GPU communication.

2. The Autoregressive Bottleneck

2.1 Why Decoding Is Slow

During the decode phase, each forward pass through a large model processes only a single new token. The GPU must read the full weight matrix from memory for every token, but performs very little arithmetic relative to the data moved. This makes decode memory-bandwidth-bound rather than compute-bound—the GPU's floating-point units sit largely idle. On a distributed system, the problem compounds: each step also triggers collective communication (all-reduce operations for tensor parallelism, or point-to-point transfers for pipeline parallelism).

2.2 The Opportunity

The target model's prefill phase, by contrast, processes many tokens in parallel and is compute-bound. If a batch of K candidate tokens could be verified simultaneously, the cost would be similar to a single forward pass (or at most a modest multiple). Speculative decoding exploits this asymmetry: draft cheaply, verify in bulk.

3. How Speculative Decoding Works

3.1 The Draft-Then-Verify Loop

The basic speculative decoding loop operates as follows. First, the draft model generates K candidate tokens autoregressively. This is fast because the draft model is small—typically 10–50x fewer parameters than the target. Second, the target model runs a single forward pass over all K candidates in parallel, producing a probability distribution at each position. Third, a verification step compares the draft model's token choices against the target model's distributions. Tokens are accepted from left to right as long as they pass a stochastic acceptance criterion. The first rejected position is resampled from an adjusted distribution, and all subsequent draft tokens are discarded.

3.2 The Acceptance Criterion

The standard acceptance criterion ensures that the output distribution is mathematically identical to what the target model would produce on its own. For each position i, the draft token is accepted with probability min(1, p_target(x_i) / p_draft(x_i)), where p_target and p_draft are the respective probabilities assigned to the drafted token. If rejected, a corrected token is sampled from a modified distribution that accounts for the rejected mass. This guarantees that speculative decoding is lossless—the output distribution is provably unchanged.

3.3 Expected Speedup

The speedup depends on the acceptance rate α, which is the average probability that a draft token is accepted. If α is high (e.g., 0.8–0.9 for fluent natural language), the expected number of accepted tokens per verification step is Kα / (1 - α^K), which can yield 3–5 tokens per target-model call with K=8 and α=0.85. The net speedup is roughly (accepted tokens per step) / (1 + cost_ratio), where cost_ratio is the relative cost of the draft model's K steps compared to one target-model step.

4. Draft Model Strategies

4.1 Independent Small Models

The most straightforward approach uses a separate, smaller model from the same family as the draft. For example, pairing Llama 3 70B as the target with Llama 3 8B as the draft. The draft model runs on a single small GPU while the target is distributed across many. The key requirement is that the draft model's vocabulary and tokenizer match the target's, so that token-level comparisons are valid.

4.2 Self-Drafting and Layer Skipping

Self-speculative decoding eliminates the need for a separate draft model entirely. Instead, the target model itself is used in a degraded mode—skipping intermediate layers, using early exit from a middle layer, or applying a smaller subset of attention heads—to produce fast but approximate predictions. This avoids the memory overhead of loading a second model and can achieve acceptance rates of 0.7–0.85, since the draft shares the same representations.

4.3 Medusa and Multi-Head Drafting

Medusa adds small prediction heads on top of the target model that predict tokens at future positions (position +1, +2, +3, etc.) in parallel from the same hidden state. These heads are lightweight (a few linear layers each) and generate candidate continuations without any autoregressive draft steps. The target model then verifies a tree of candidates in a single forward pass using tree attention. Medusa can achieve 2–3x speedup with negligible memory overhead, making it particularly attractive for small-GPU deployments where there is no room for a separate draft model.

4.4 Prompt Lookup and N-gram Drafting

For workloads where the output is likely to repeat content from the input—such as code editing, document reformatting, or summarization with extraction—the draft can be generated by simply looking up n-gram matches in the prompt. This requires zero additional model parameters and zero additional computation. The acceptance rate is workload-dependent but can be very high (>0.9) for copy-heavy tasks.

5. Speculative Decoding on Distributed Small-GPU Systems

5.1 Communication Reduction

On a system where the target model is tensor-parallel across N small GPUs, each decode step triggers 2L all-reduce operations (where L is the number of transformer layers). If speculative decoding accepts an average of 3 tokens per verification step, the number of all-reduce rounds drops by roughly 3x. For a 70B model on 8 GPUs connected via PCIe, where communication can consume 60–70% of decode time, this translates to a substantial wall-clock improvement.

5.2 Draft Model Placement

The draft model should ideally run on a dedicated GPU that is not part of the target's tensor-parallel group, so that draft generation and target verification can be pipelined. If no spare GPU is available, the draft model can share a GPU with one shard of the target model, running during the intervals when that GPU would otherwise be idle (e.g., during pipeline bubbles). Alternatively, self-drafting or Medusa heads avoid the placement question entirely.

5.3 Batch Interaction

Speculative decoding interacts with continuous batching in subtle ways. Different sequences in a batch may have different acceptance rates, causing them to advance by different numbers of tokens per step. Serving frameworks like vLLM and TensorRT-LLM handle this by padding the verification batch and masking out sequences that have already been resolved for that step. The efficiency of speculative decoding generally improves at smaller batch sizes, where the decode phase is most memory-bandwidth-bound.

6. Practical Considerations

- **Acceptance rate varies by domain:** Code, structured data, and formulaic text see acceptance rates of 0.85–0.95. Creative writing and reasoning-heavy tasks may drop to 0.5–0.7, reducing the benefit.
- **Draft quality matters more than draft speed:** A slightly slower draft model with a higher acceptance rate often outperforms a faster but less accurate one, because each rejected token wastes an entire target-model verification slot.

- **Tree-based verification amplifies throughput:** Instead of verifying a single linear sequence, the system can verify a tree of candidate continuations. This increases the probability that at least one path is accepted to a greater depth, at the cost of a wider (but still single) forward pass.
- **Quantization synergy:** The draft model can be aggressively quantized (e.g., INT4) since its predictions only need to be approximately correct. Combined with a quantized target model, this further reduces memory and communication pressure on small GPUs.

7. Conclusion

Speculative decoding is one of the most impactful inference optimizations available for distributed small-GPU systems. By reducing the number of expensive, communication-heavy target-model forward passes by 2–3x, it directly addresses the primary bottleneck of autoregressive generation. The technique is lossless when using the standard acceptance criterion, requires no retraining of the target model (for independent draft or n-gram approaches), and composes well with other optimizations like quantization, PagedAttention, and continuous batching. For any practitioner deploying large models on resource-constrained hardware, speculative decoding should be among the first techniques evaluated.