Speculative Decoding

INTRODUCTION TO LLM INFERENCE SERVING SYSTEMS

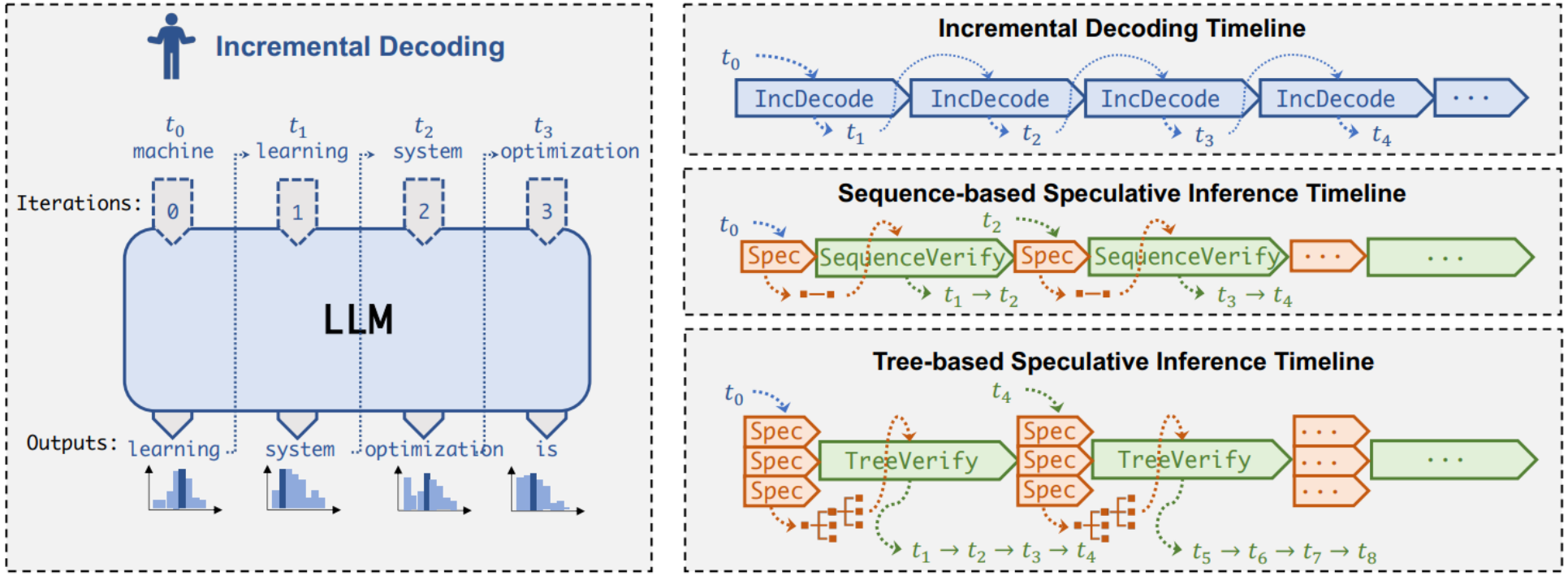
CHUHONG YUAN

Overview

- SpecInfer
- Discussion
- EAGLE
- Homework Review

SpecInfer

Background



(a) Incremental decoding.

(b) Timeline Comparison.

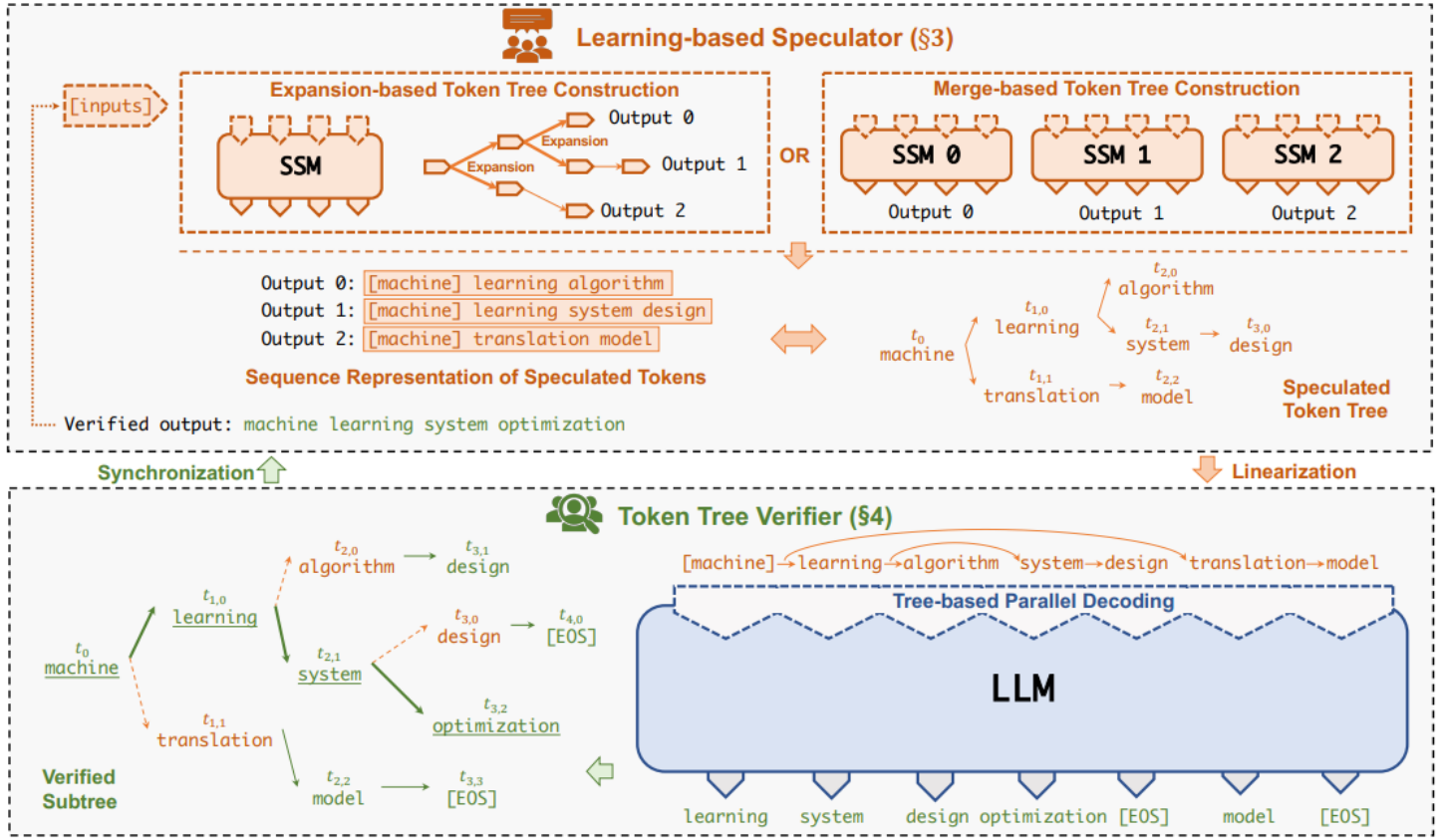
Challenges

- The search space is large
- Verifying stochastic decoding

Challenges

- The search space is large
  - Small speculative models (SSM)
  - Expansion- and merge-based mechanism for constructing token trees
- Verifying stochastic decoding
  - Multi-step speculative sampling
  - Tree-based parallel decoding

SpecInfer Overview



Learning-based Speculator

- Tree-based speculation
  - Provide more token candidates for each step

Learning-based Speculator

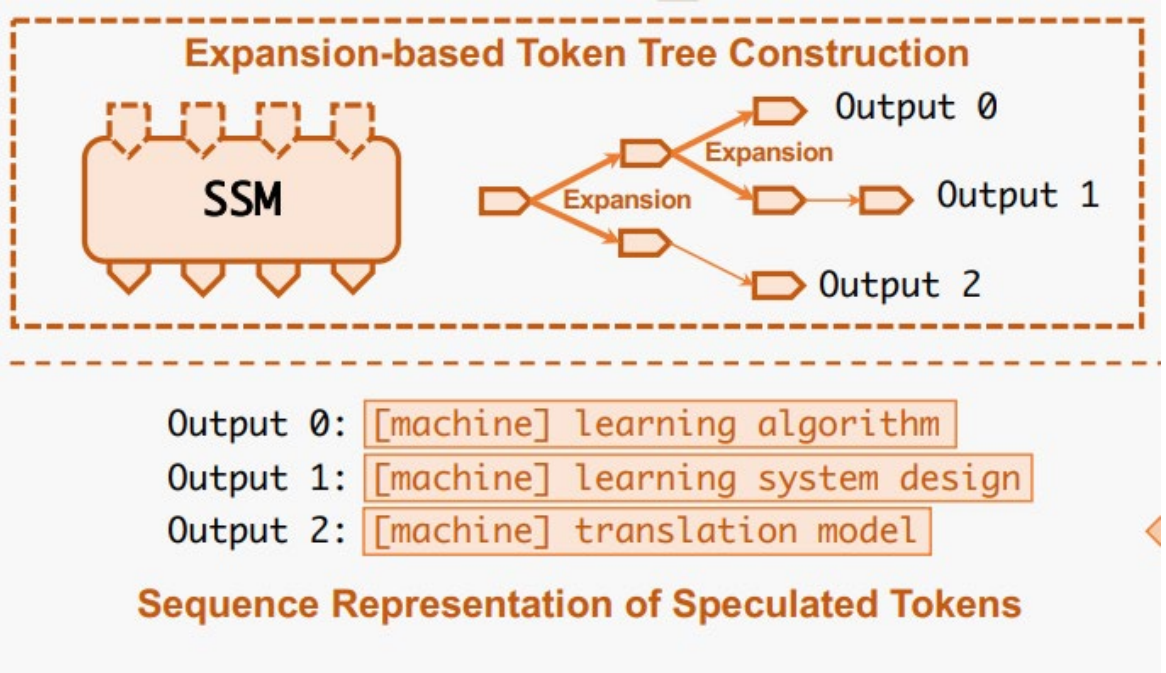
- Tree-based speculation
  - Provide more token candidates for each step

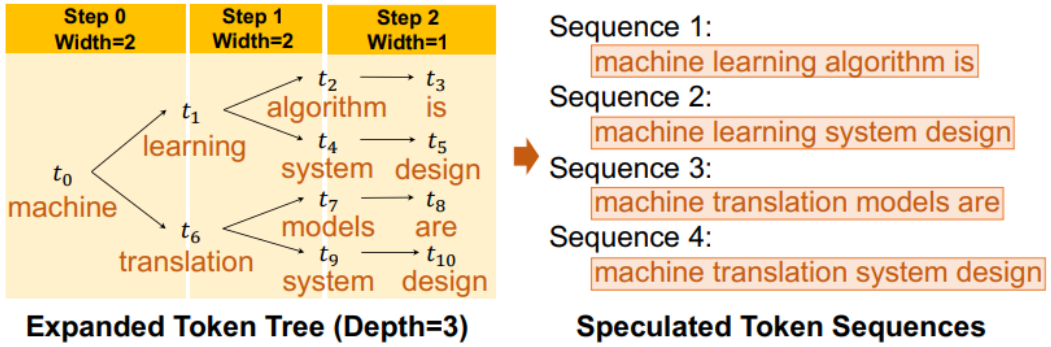
	Dataset	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
Greedy decoding	Alpaca	68%	77%	81%	84%	85%
	CP	69%	79%	83%	86%	87%
	WebQA	62%	72%	77%	80%	82%
	CIP	70%	81%	85%	88%	89%
	PIQA	63%	75%	79%	83%	85%
Stochastic decoding	Alpaca	54%	81%	91%	95%	97%
	CP	56%	82%	92%	95%	97%
	WebQA	52%	80%	90%	94%	96%
	CIP	57%	84%	92%	95%	97%
	PIQA	55%	82%	91%	95%	97%

Learning-based Speculator

- Tree-based speculation
  - Provide more token candidates for each step
- Expansion-based token tree construction
  - One SSM generates multiple candidates for each step
- Merge-based token tree construction
  - Multiple SSMs, each one generates a token tree and the trees are merged

Expansion-Based Token Tree Construction



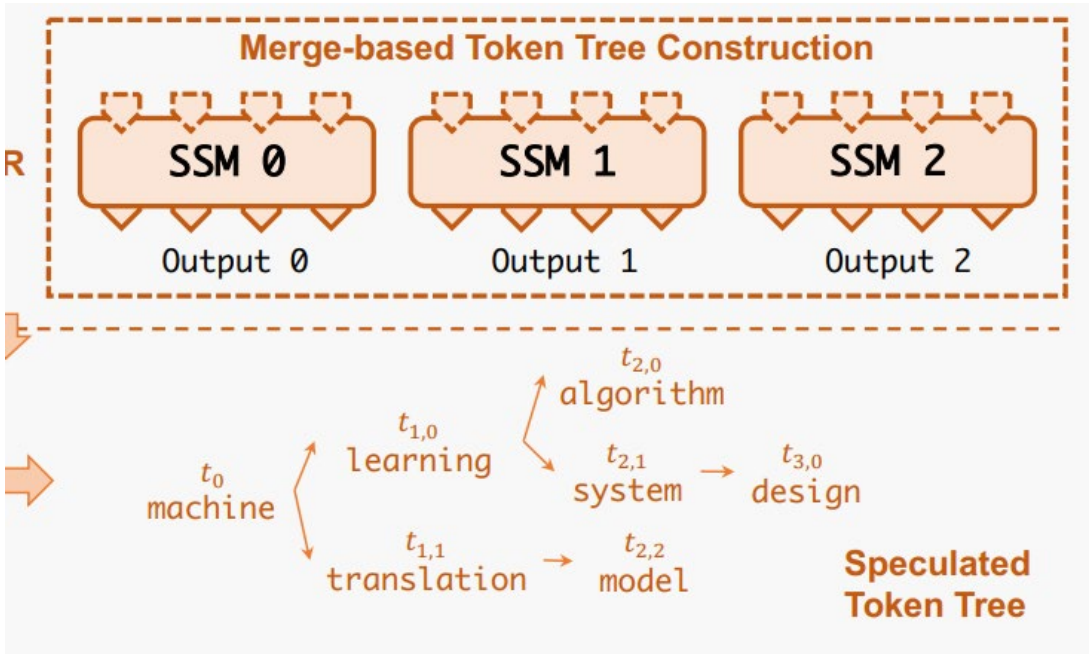


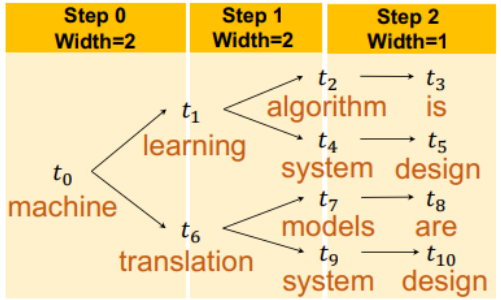
Statically set the number of tokens at each step to <2,2,1>

Merge-Based Token Tree Construction

- A pool of SSMs are tuned to align with the LLM
  - Use a collection of text prompts
  - Each time fine-tune one SSM to the fullest and mark all the prompts where the outputs are identical
  - Filter out the marked samples and use the remaining ones to fine-tune the next SSM
  - Diverse but the aggregated outputs align with the LLM

Merge-Based Token Tree Construction



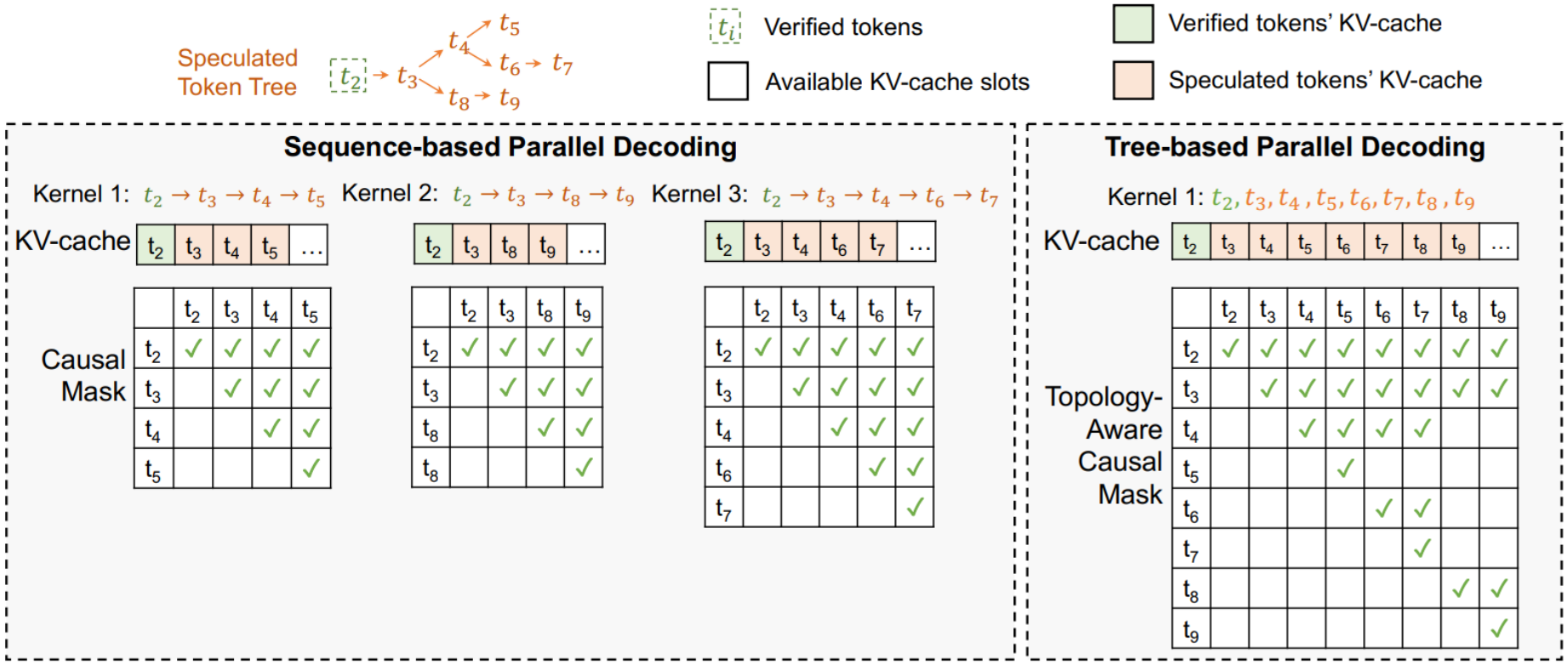


Expanded Token Tree (Depth=3)

Sequence 1:
machine learning algorithm is
Sequence 2:
machine learning system design
Sequence 3:
machine translation models are
Sequence 4:
machine translation system design

Speculated Token Sequences

Token Tree Verifier

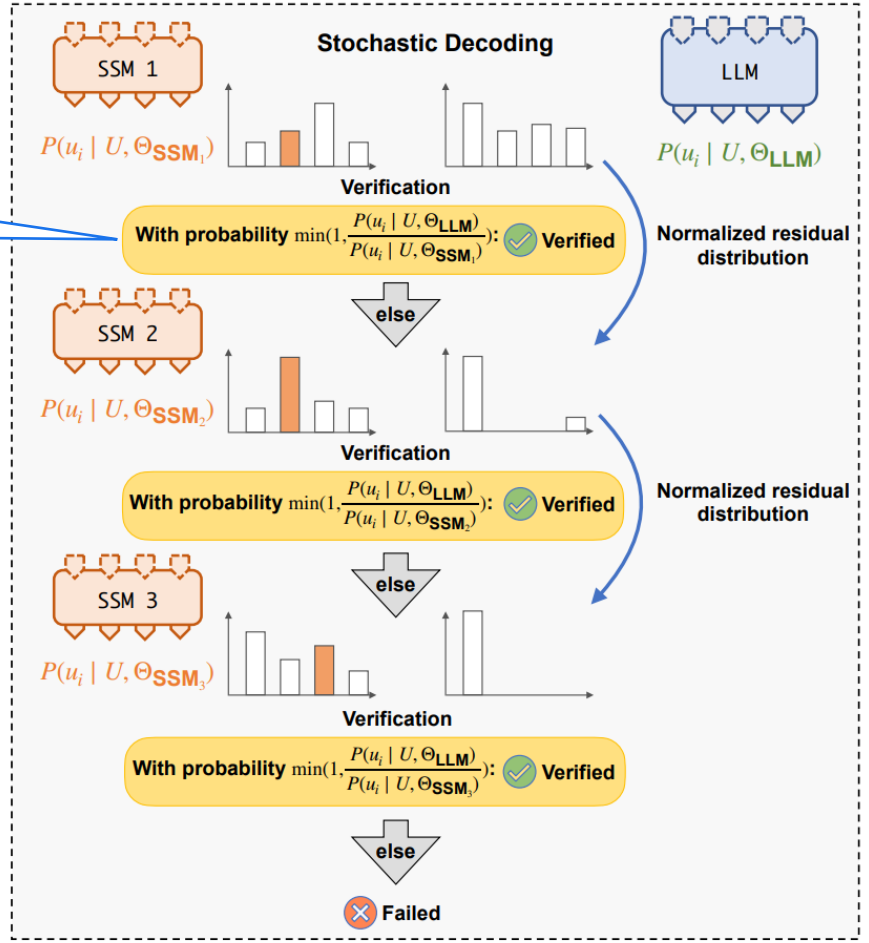


Token Tree Verifier

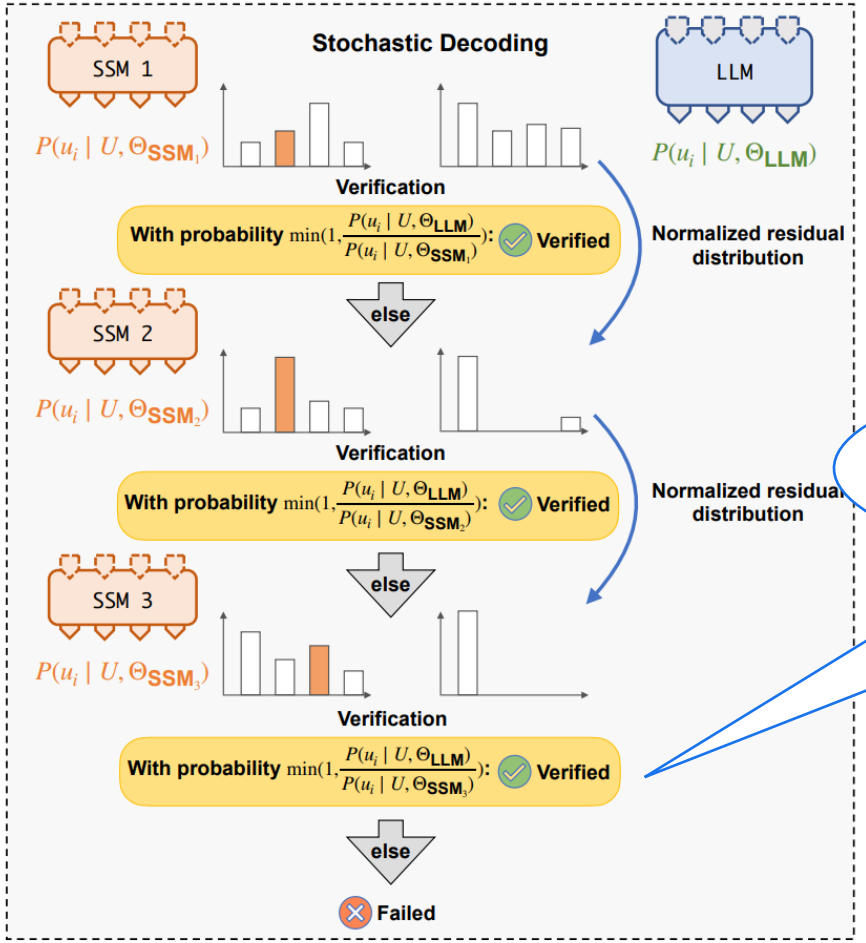
- Greedy decoding
  - Each time, only the token with the highest probability is selected
  - Just need to verify the tokens in a sequence one by one
- Stochastic decoding
  - The tokens are selected randomly based on the probability distribution
  - Need to guarantee that the speculated tokens' probability distribution remains the same
  - Proof: https://scispace.com/pdf/specinfer-accelerating-generative-llm-serving-with-2zam7dga.pdf

Multi-Step Speculative Sampling

By comparing with a random number r ~ U(0,1)

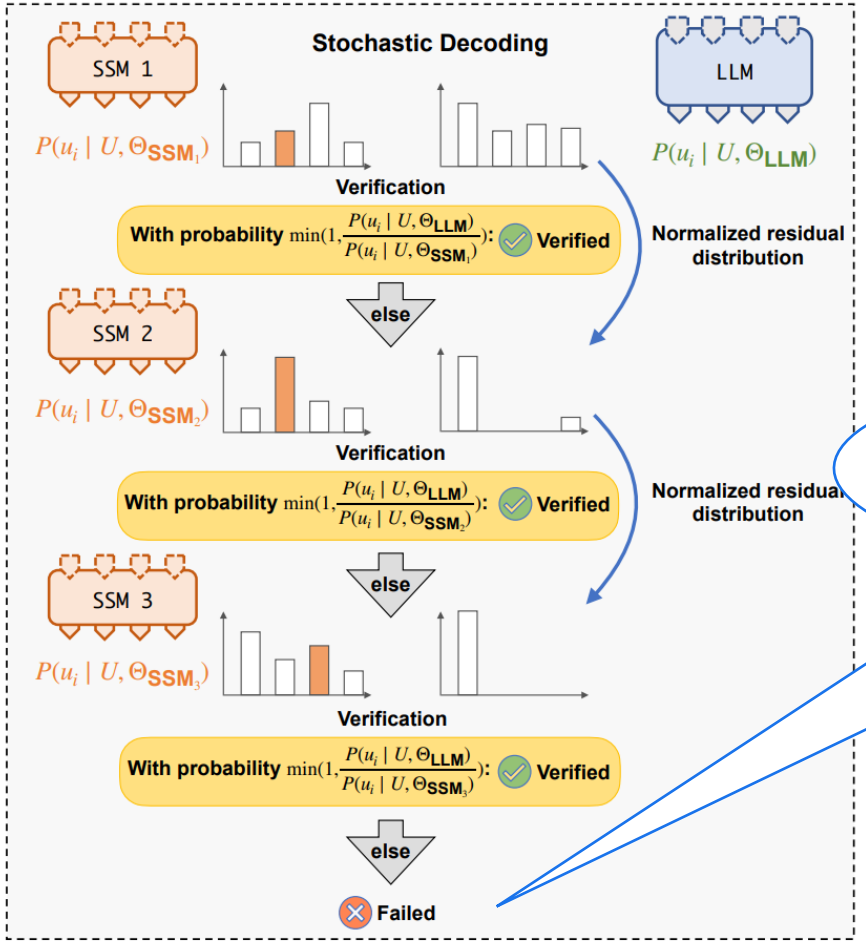


Multi-Step Speculative Sampling



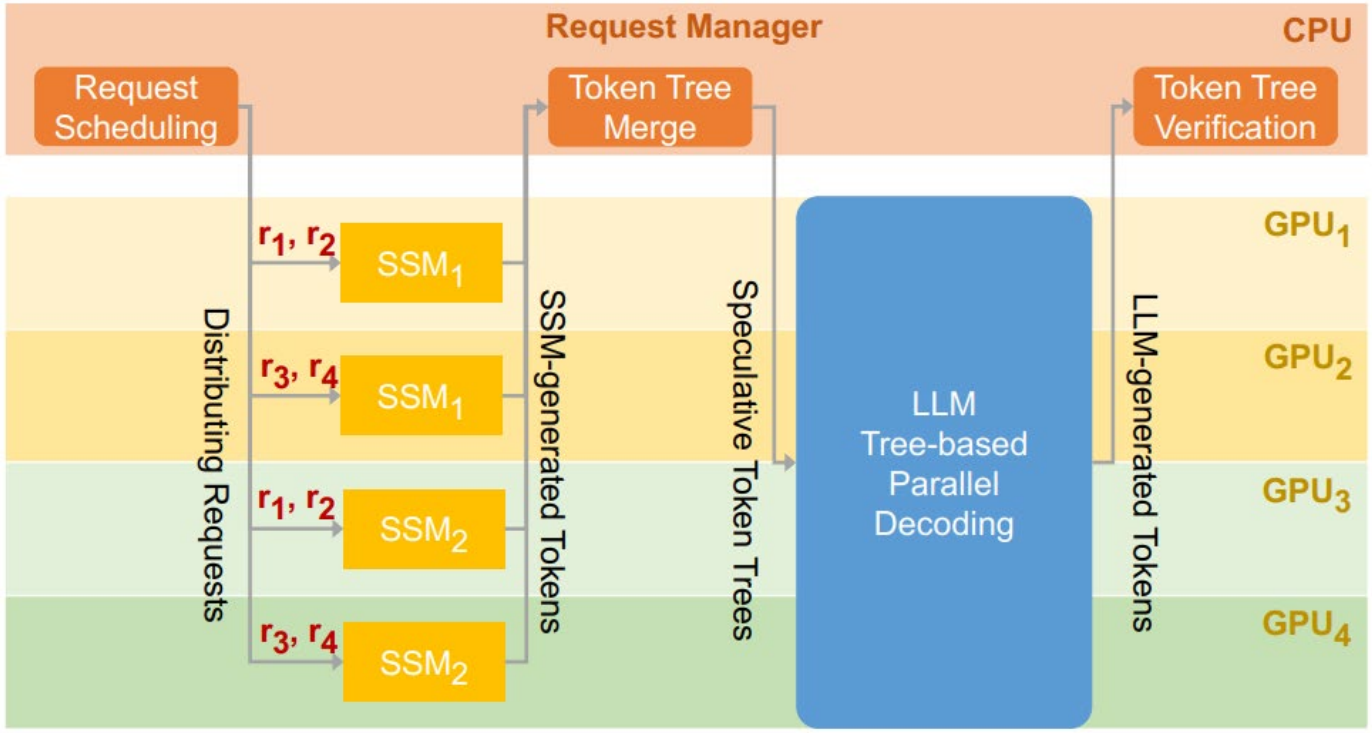
$$P\left(u_i \mid U\,;\,\Theta_{LLM}\right) = P_{SpecInfer}\left(u_i \mid U\,;\Theta_{LLM}, \{\Theta_{SSM_j}\}\right)$$

Multi-Step Speculative Sampling



$$P\,(reject \mid MSS) \;\leq\; P\,(reject \mid NS)$$

System Design



Overhead Analysis

- Memory overhead
  - SSMs, much smaller than the LLM
  - Token candidates, fewer than the input length
- Computation overhead
  - SSM computation
  - Tree attention

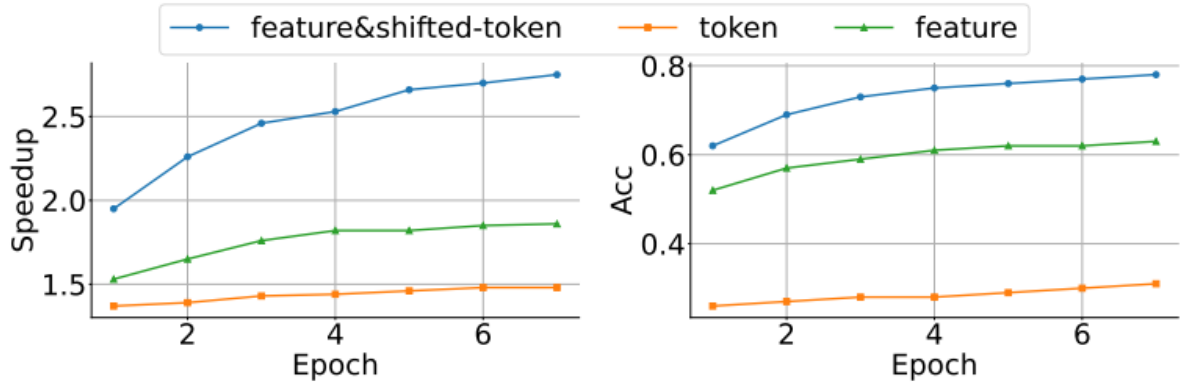
Discussion

- Limitations of SpecInfer?
- Scenarios where SpecInfer should be applied?

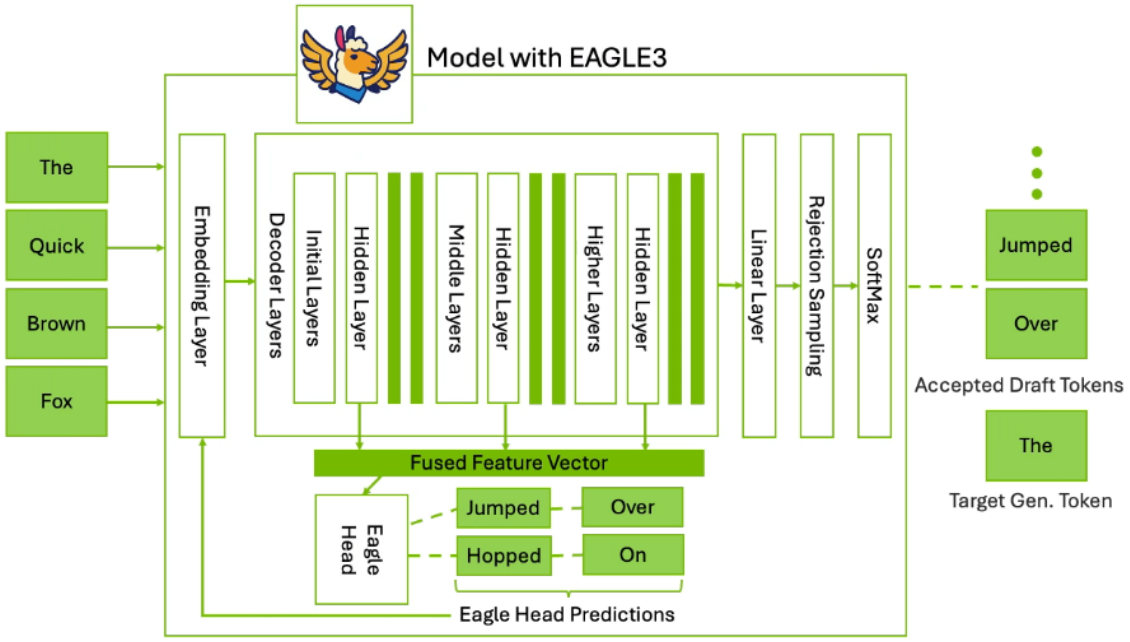
EAGLE

Motivation

- Need for SSM
  - Extra memory cost
  - Extra training cost
- Using tokens as inputs is not efficient

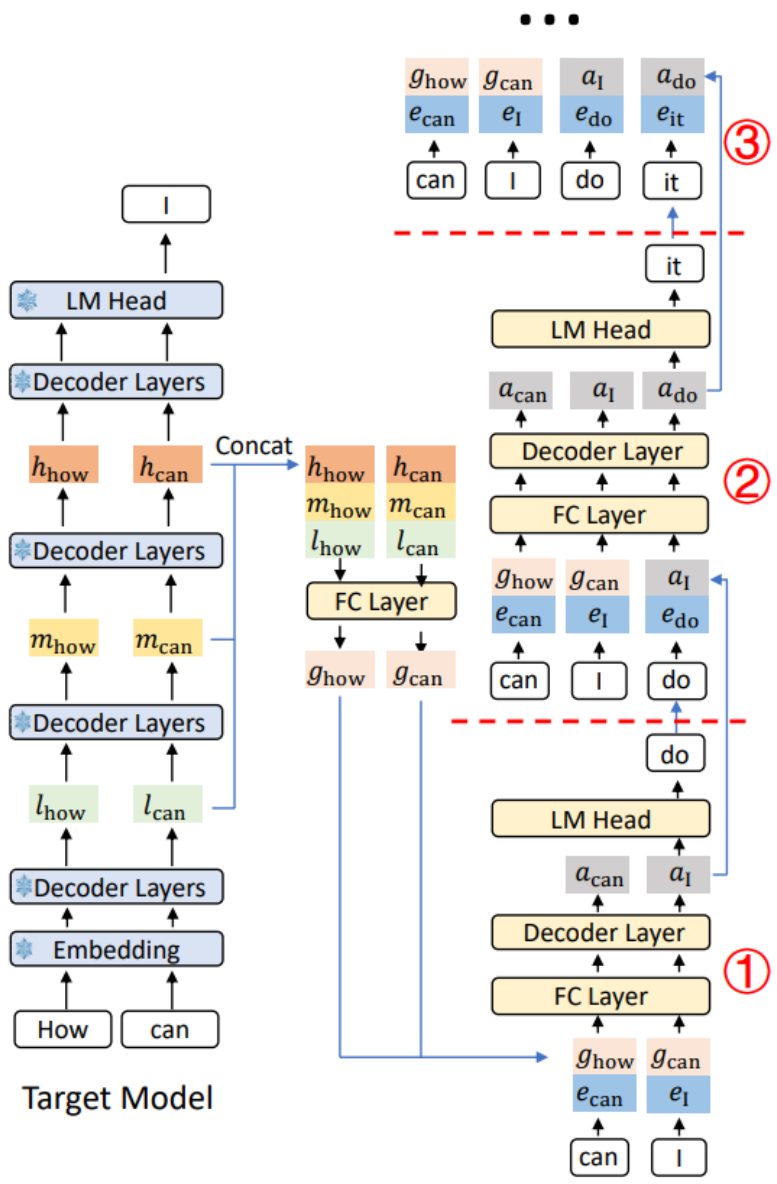


EAGLE3 Design



- Embedding layer and LM Head are reused
- Decoding layer is lightweight
- Reference: https://developer.nvidia.com/blog/an-introduction-to-speculative-decoding-for-reducing-latency-in-ai-inference/
- https://docs.vllm.ai/en/v0.10.1/features/spec_decode.html

EAGLE3 Inference Pipeline



Homework Review

- Evaluation
  - LLMs: LLaMA-7B, OPT13B, OPT-30B, and LLaMA-65B
  - SSMs: LLaMA-68M and OPT-125M
  - Datasets: Chatbot Instruction Prompts (CIP), ChatGPT Prompts (CP), WebQA, Alpaca, and PIQA
  - Platform: NVIDIA A10 24GB GPUs, 48 CPU cores, and 192 GB DRAM
  - Metrics: TPOT, verified tokens per step
  - Inference latency, token tree width, tree-based parallel decoding, multi-step speculative sampling

Homework

- Read the related materials, and answer the following questions:
  - 1. What are the major features of Dynamo?
  - 2. How does Dynamo select the best-matched prefill worker?
  - 3. What are the evaluation workloads and metrics used in Nvidia's study "Beyond the Buzz"?
  - 4. What factors are included in the conditional disaggregation of Dynamo? Based on the research results of Nvidia's study, why are these factors considered?
  - 5. Critical thinking: Does Nvidia's study completely demonstrate the conditions where P-D disaggregation can be beneficial? What experiments should be added?
- Related materials:
  - Dynamo official website: https://www.nvidia.com/en-us/ai/dynamo/
  - Dynamo Git Repo: https://github.com/ai-dynamo/dynamo (You may need to check components/backends/vllm/src/dynamo/vllm_prefill_router/__main__.py and lib/llm/src/disagg_router.rs)
  - Dynamo documentation: https://docs.nvidia.com/dynamo/latest/index.html
  - Beyond the Buzz paper: https://arxiv.org/html/2506.05508v1

Q & A