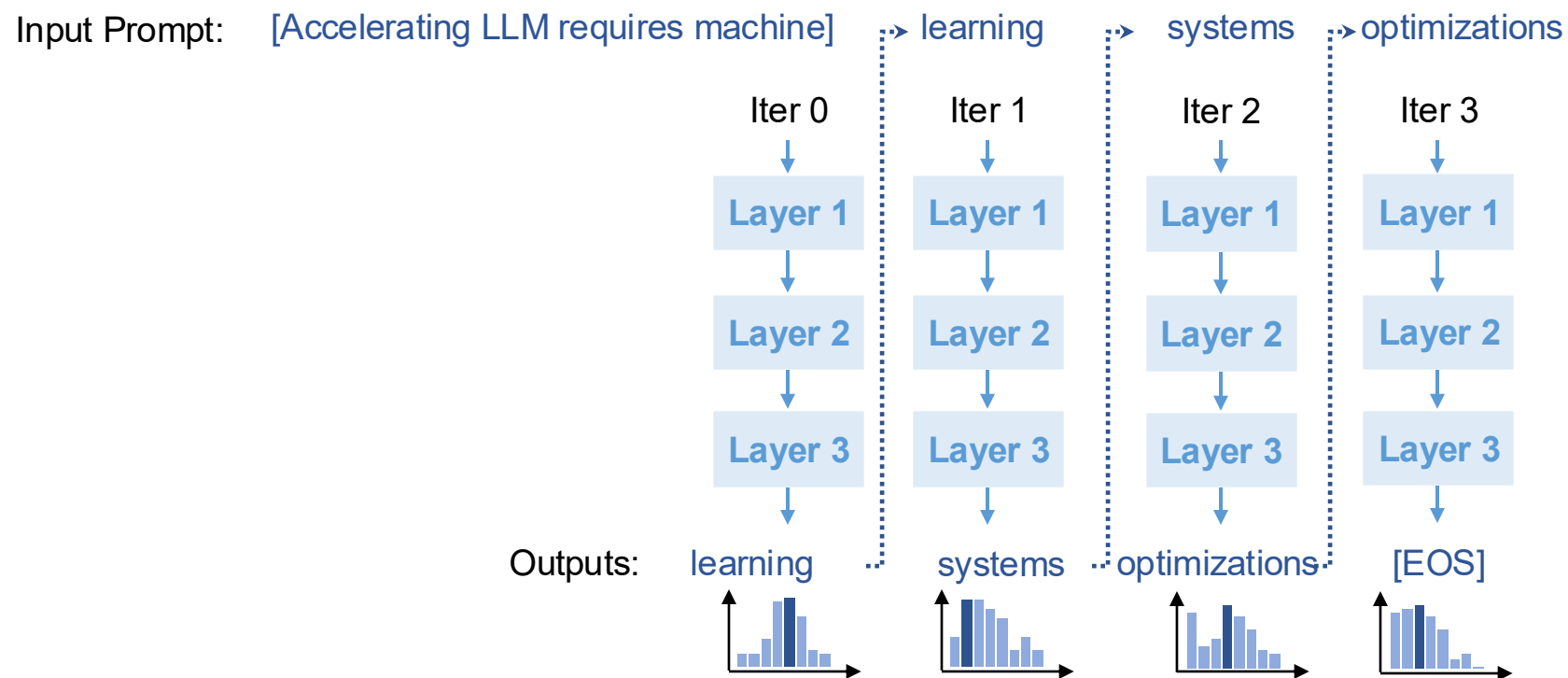


15-779: Lecture 16

LLM Serving Part 2: Speculative Decoding

Tianqi Chen and Zhihao Jia
Carnegie Mellon University

Generative LLM Inference: Autoregressive Decoding



LLMs are Slow and Expensive to Serve

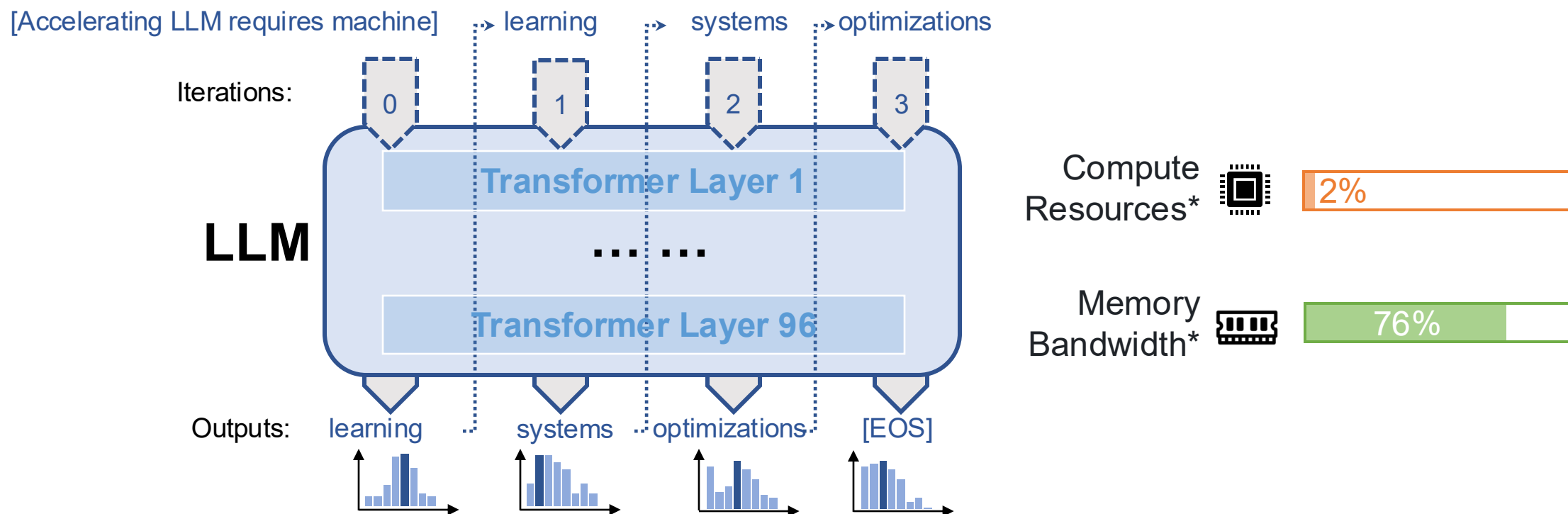


- **At least ten** A100-40GB GPUs to serve 175B GPT-3 in half precision
- Generating 256 tokens takes **~20 seconds**
- Cannot process many requests in parallel
 - Per-request key/value cache takes **3GB GPU memory**

Today's Lecture: Speculative Decoding

- **Model-based speculative decoding**
 - Using a separate draft model to predict LLM's output
- Model-free speculative decoding
 - Using previously generated tokens to predict future tokens

Recall: Incremental Decoding Issues



- Limited degree of parallelism → underutilized GPU resources
- Need all parameters to decode a token → bottlenecked by GPU memory access

Tradeoffs between Different Language Models

# Parameters	175B	13B	2.7B	760M	125M
TriviaQA	71.2	57.5	42.3	26.5	6.96
PIQA	82.3	79.9	75.4	72.0	64.3
SQuAD	64.9	62.6	50.0	39.2	27.5
latency	20 s	7.6s	2.7s	1.1s	0.3s
# A100s	10	1	1	1	1

Comparing multiple GPT-3 models*

Large models

 Pro: better generative performance

 Con: slow and expensive to serve

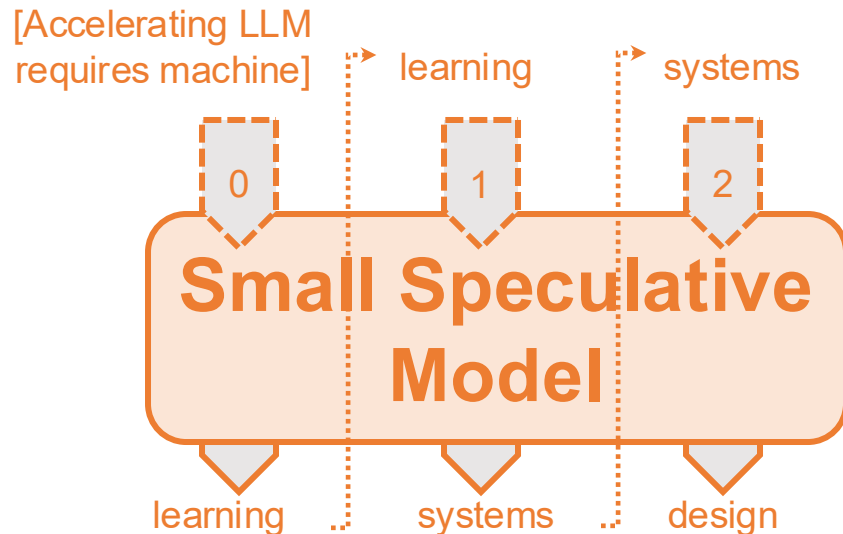
Small models

 Pro: cheap and fast

 Con: less accurate

Speculative Decoding

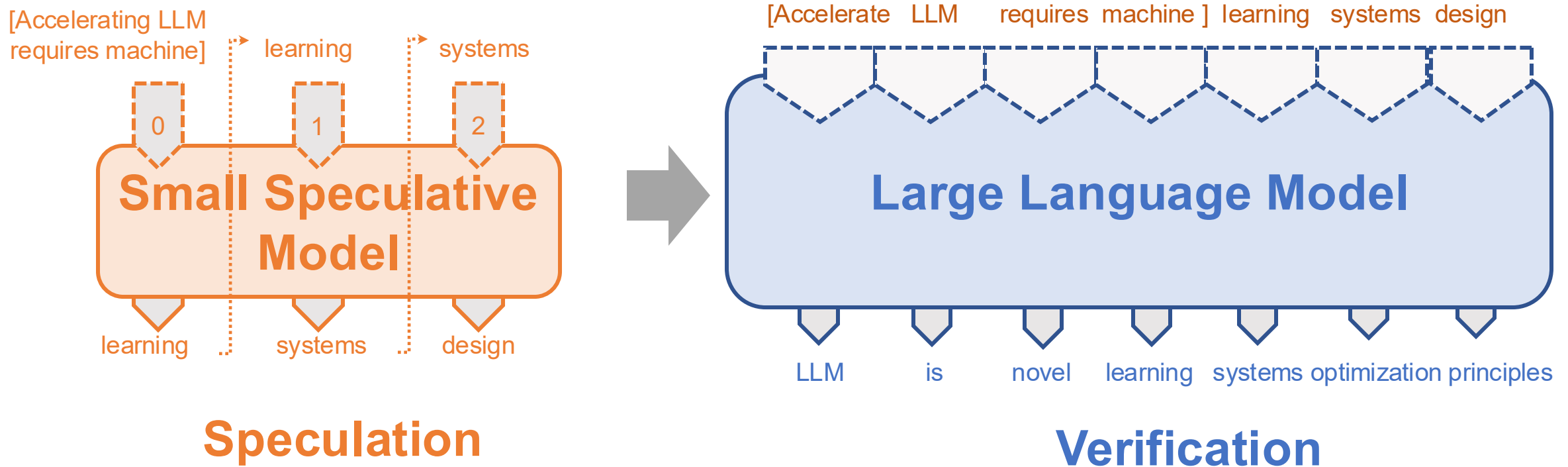
1. Use a small speculative model (SSM) to predict the LLM's output
 - SSM runs much faster than LLM



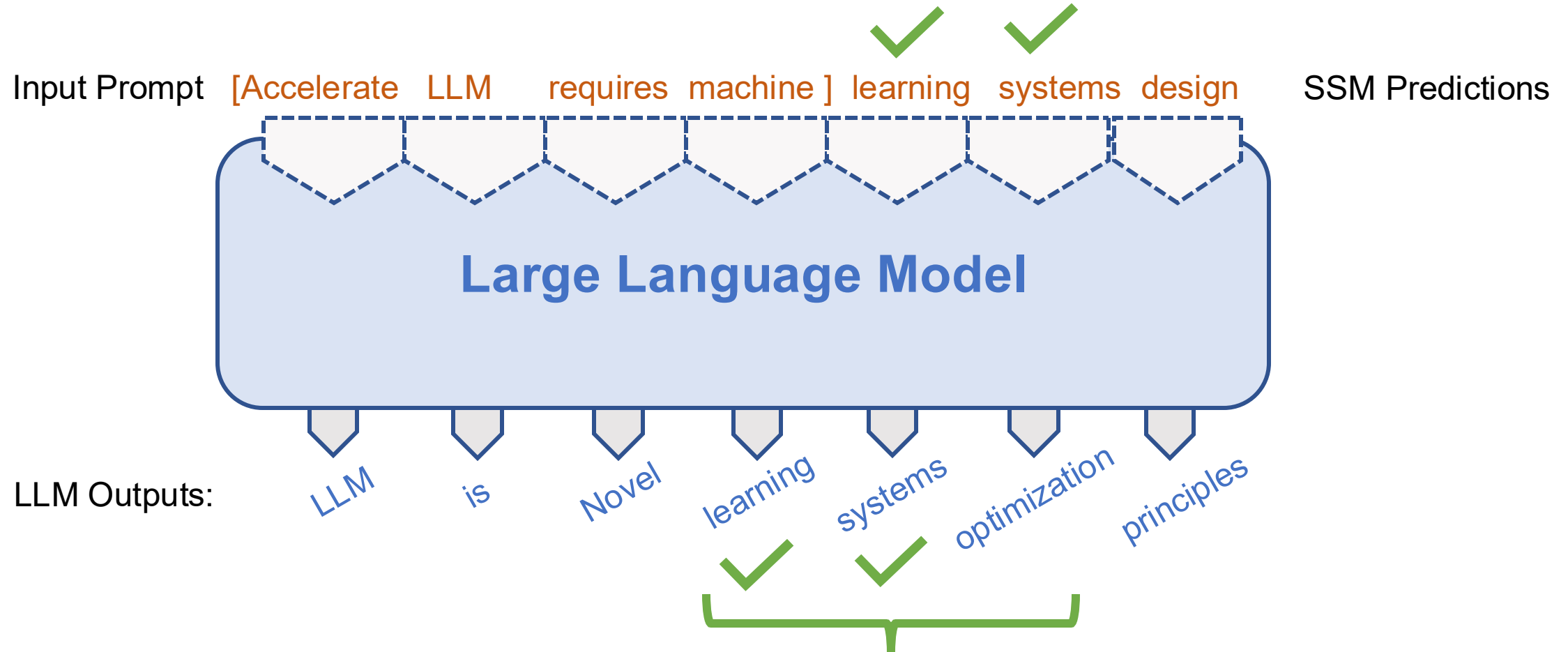
Speculation

Speculative Decoding

1. Use a small speculative model (SSM) to predict the LLM's output
 - SSM runs much faster than LLM
2. Use the LLM to verify the SSM's prediction

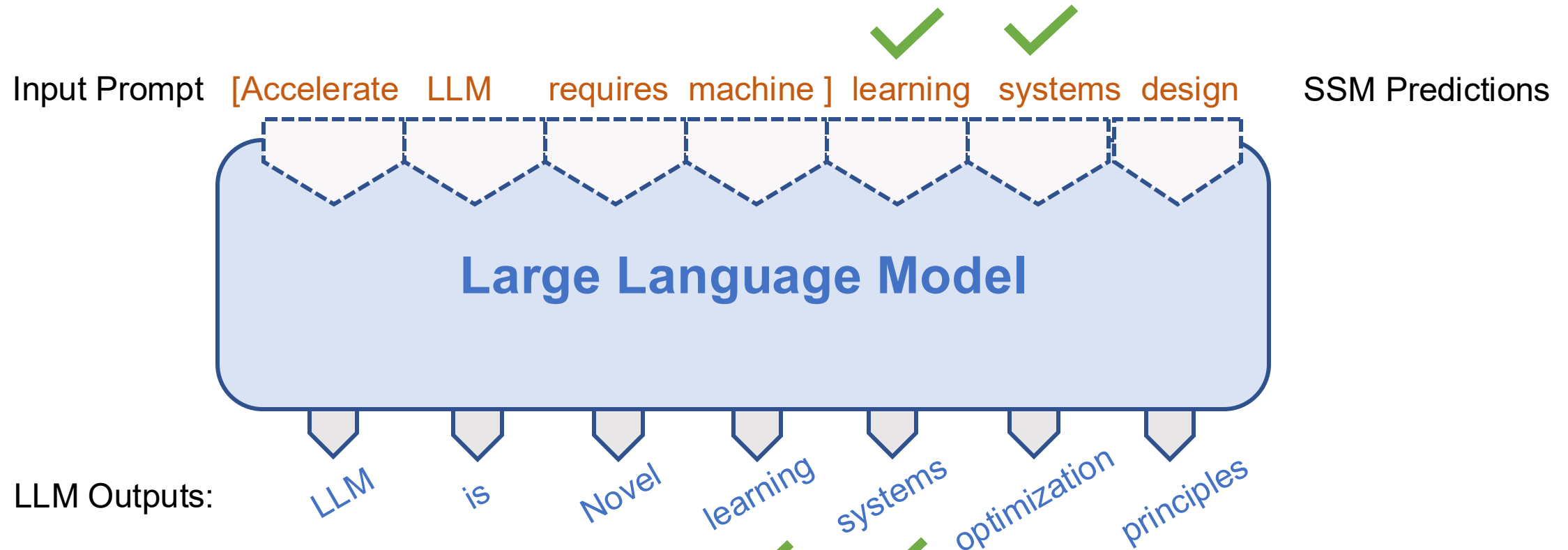


Verifying Speculative Decoding Results



Generate 3 new tokens in one LLM decoding step

Verifying Speculative Decoding Results



Key takeaway:

- LLM inference is bottlenecked by accessing model weights
- using LLM to decode multiple tokens to improve GPU utilization

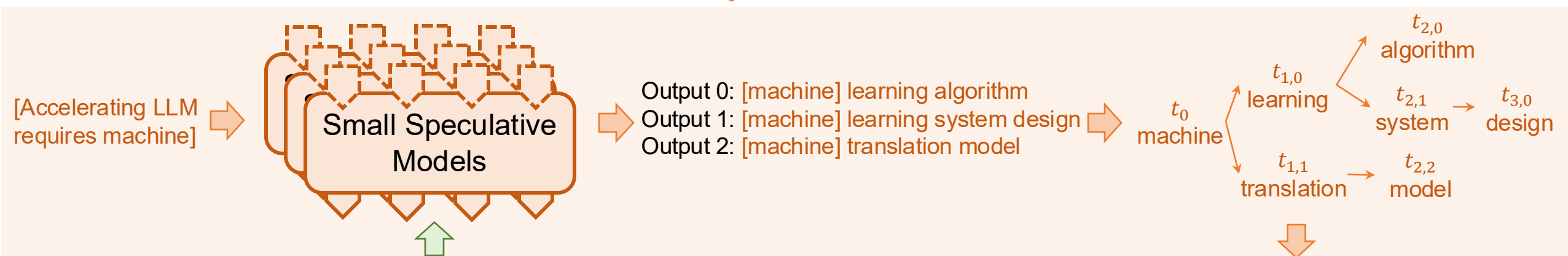
SpecInfer: Tree-based Speculative Inference & Verification

Key idea: not use LLMs as incremental decoder, use them as **parallel token tree verifier**

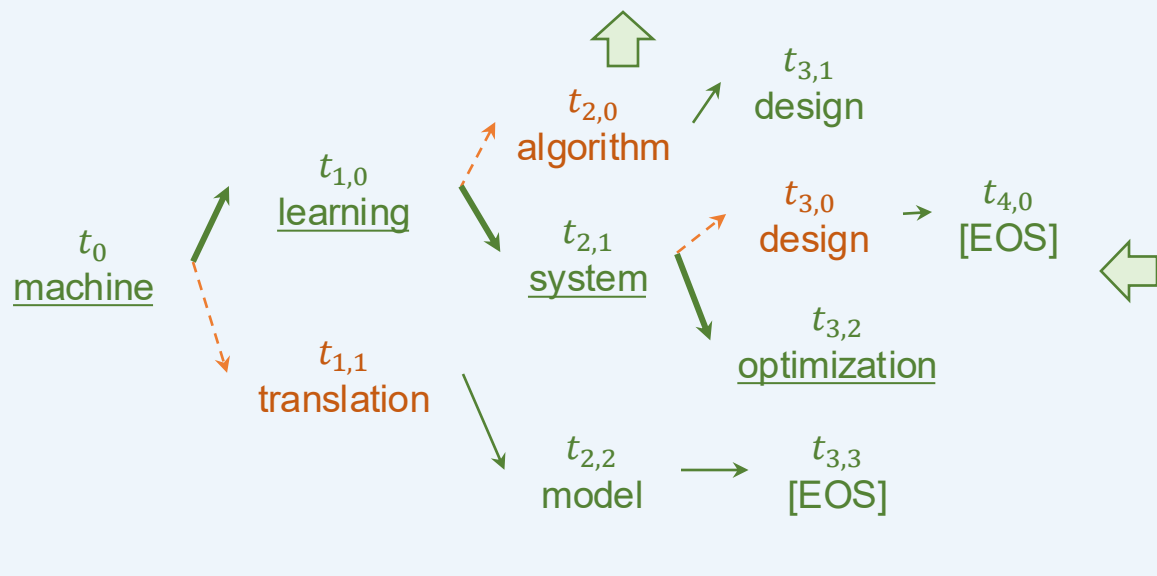
- **Better performance:** outperform existing LLM systems by 1.3-2.4x
- **Higher efficiency:** reduce GPU memory access by 2.5-4.4x
- **Correctness:** verification guarantees end-to-end equivalence

SpecInfer Workflow

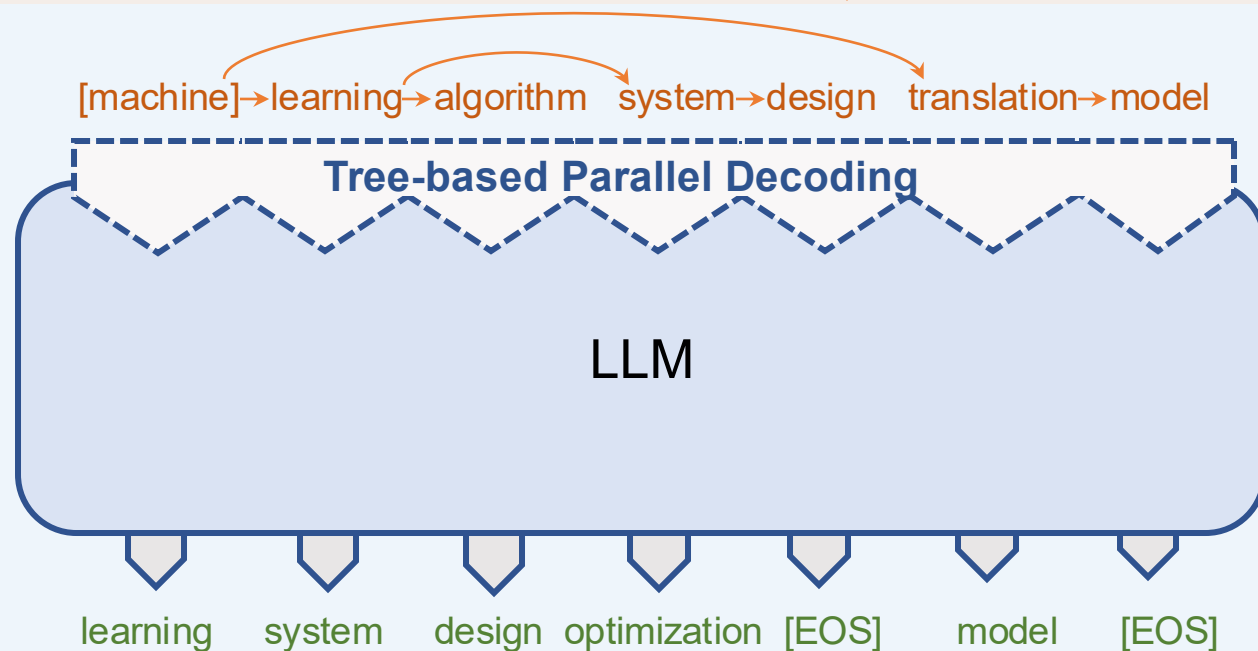
Speculator



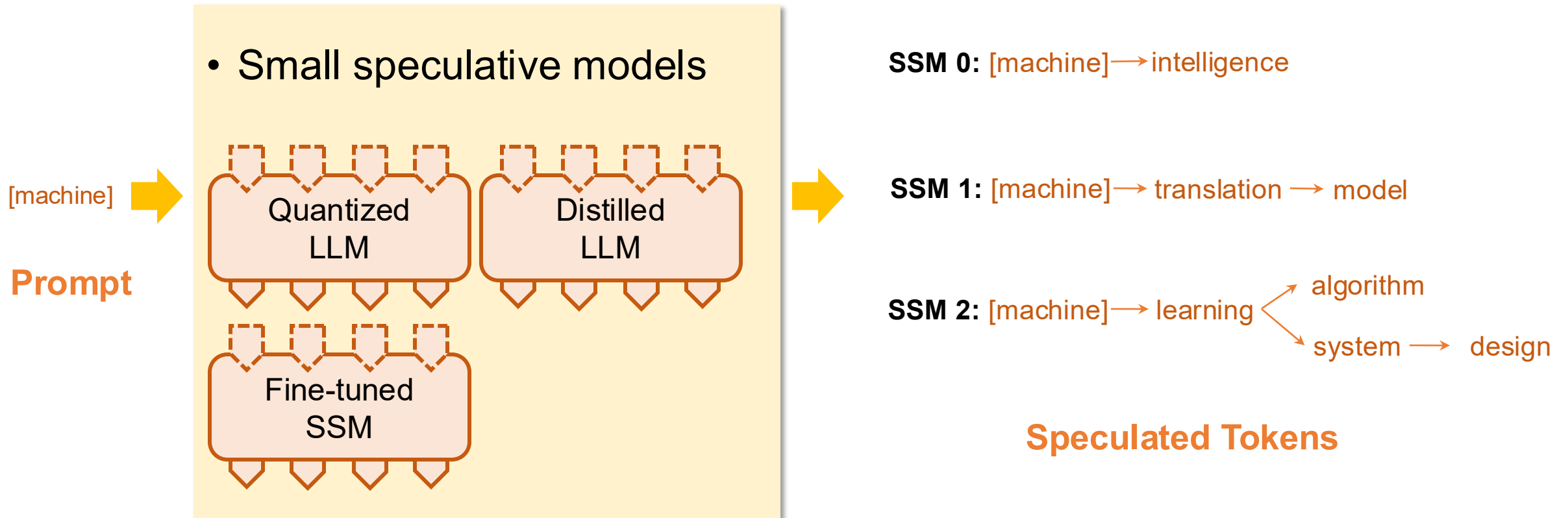
Verified output: machine learning system optimization

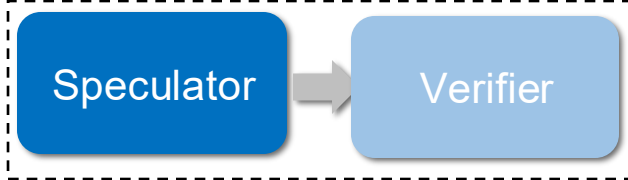


Verifier

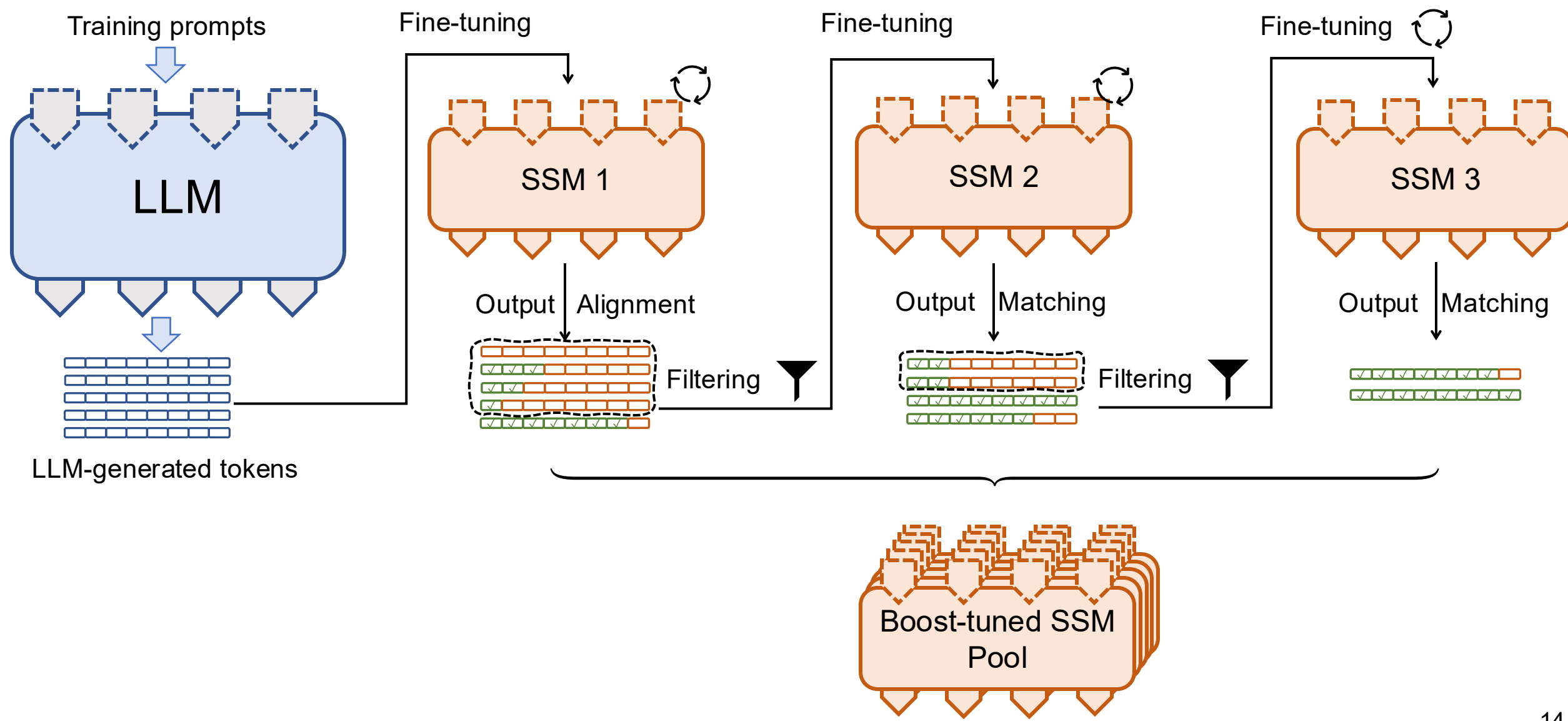


Learning-based Speculator

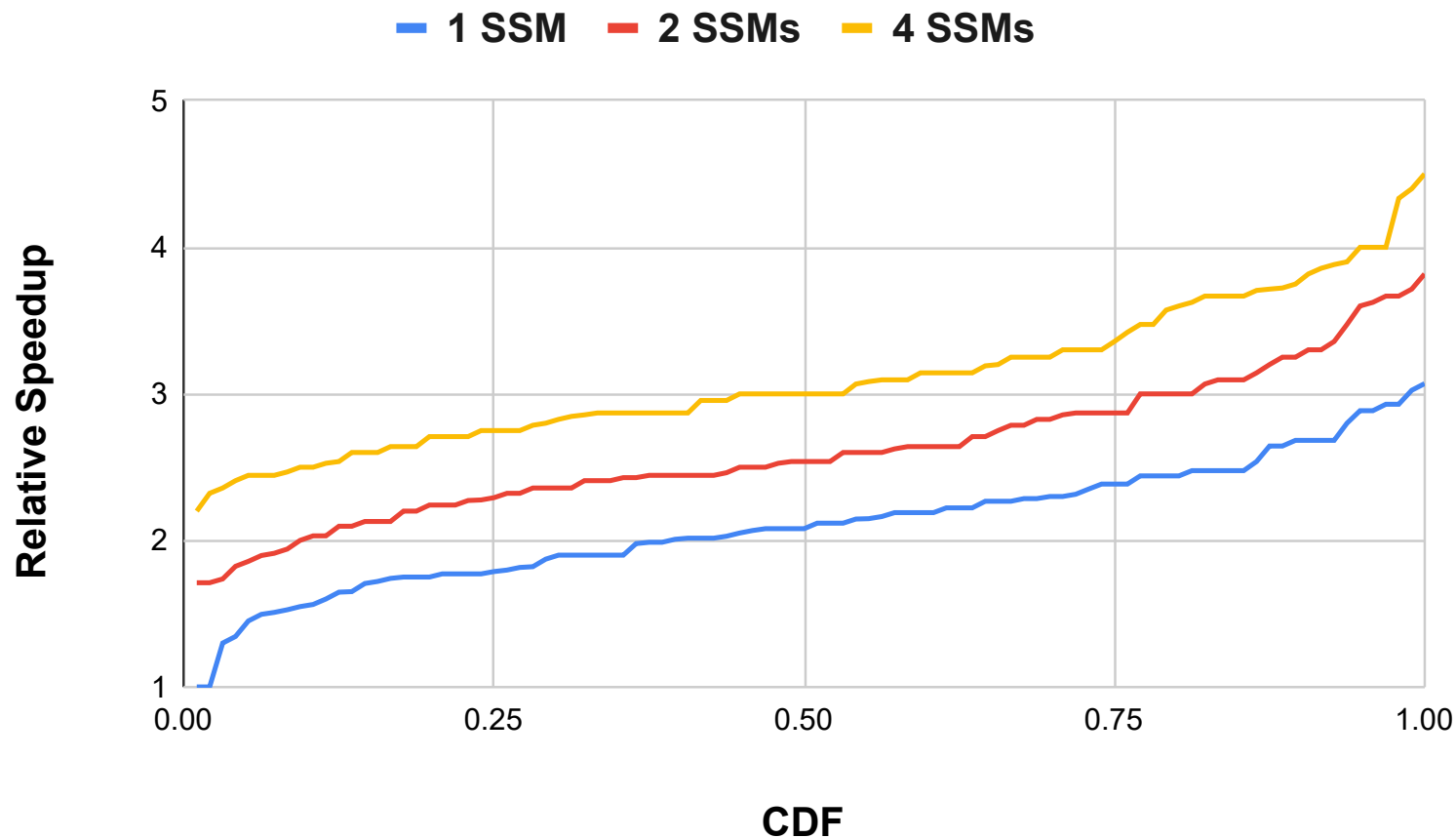


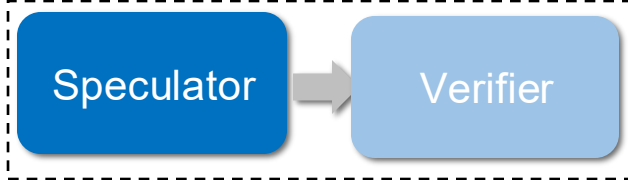


Collective Boost-Tuning



Collective Boost-Tuning Consistently Improves Performance



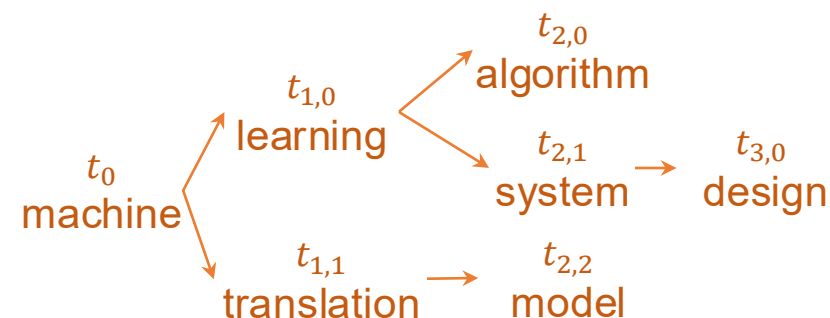
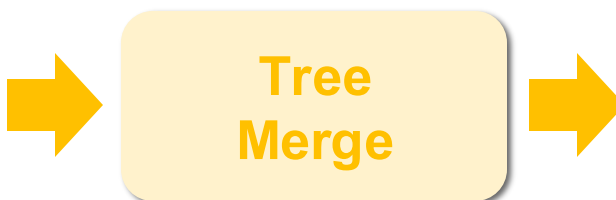


Token Tree Merge

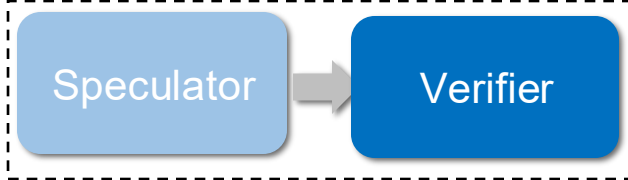
- A compact way to represent speculated tokens

SSM 0: [machine] learning algorithm
SSM 1: [machine] learning system design
SSM 2: [machine] translation model

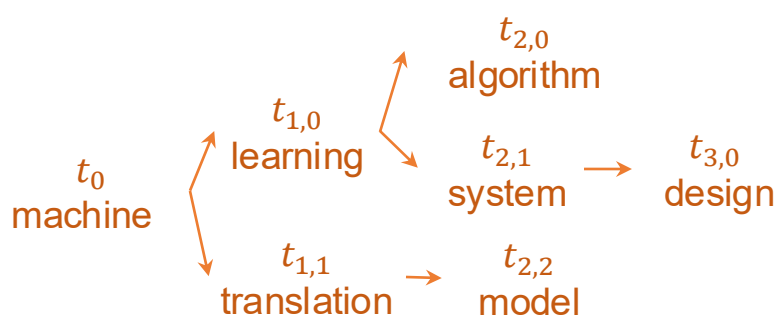
Token Sequences



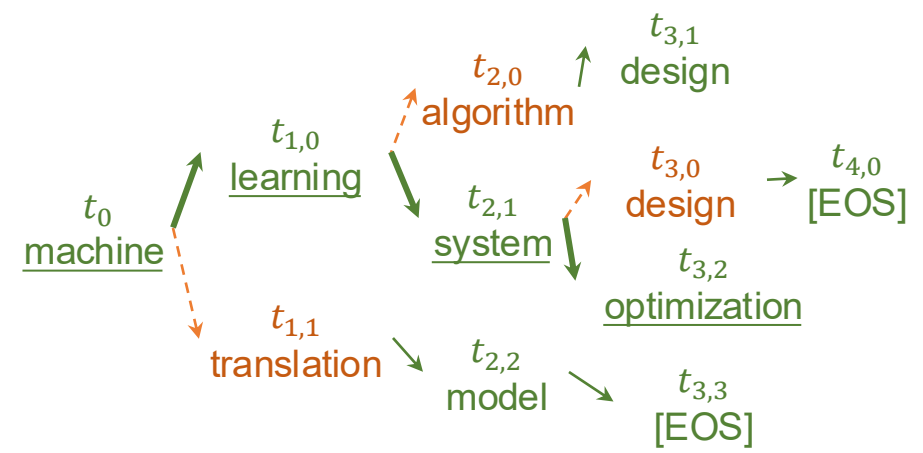
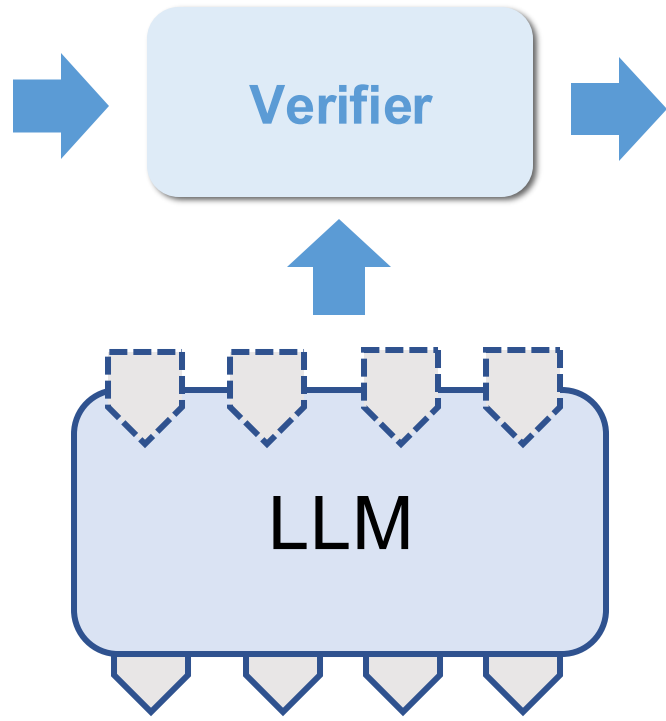
Token Tree



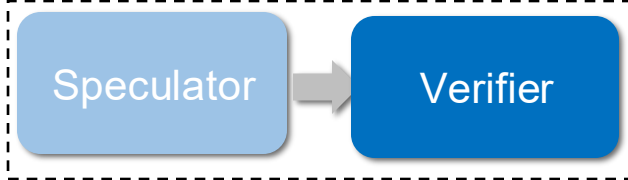
Token Tree Verifier



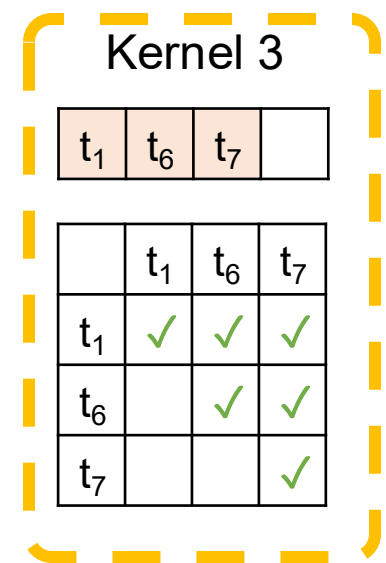
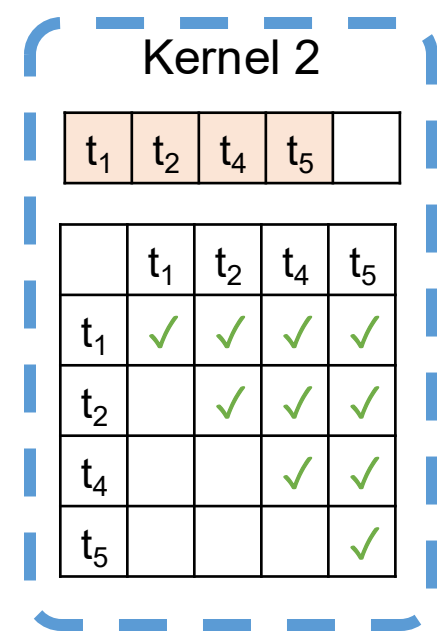
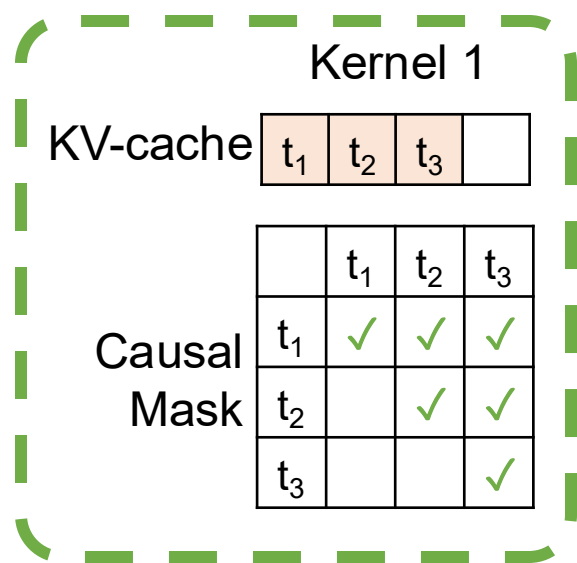
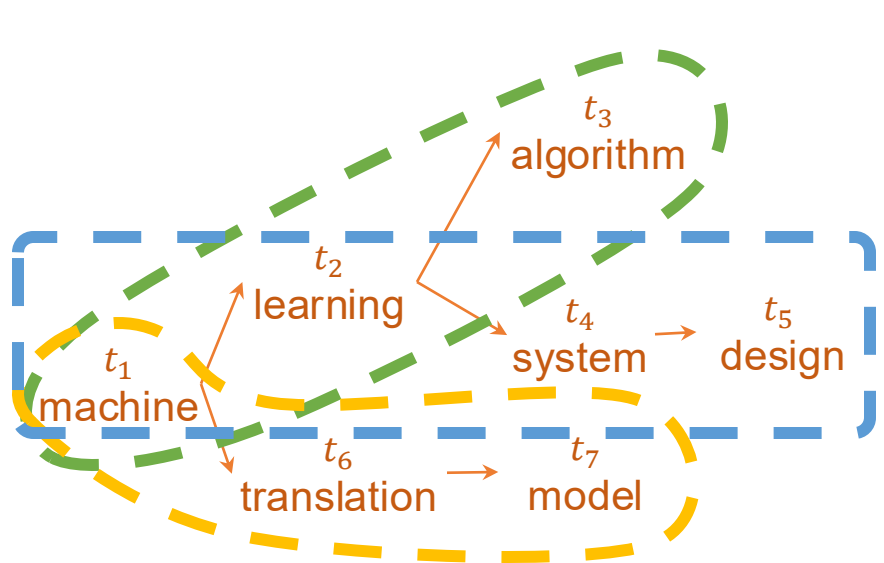
Speculated token tree



Verified token tree



Sequence-based Decoding



Issues:

- Redundant decoding computation
- More requests → more GPU memory for key/value cache

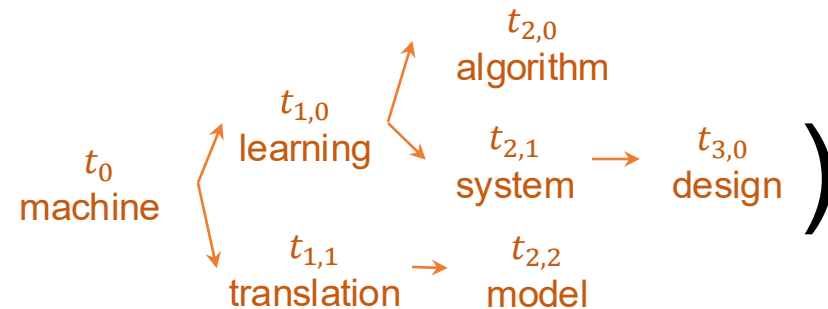
Tree Attention

- same output as sequence attention for each token; no redundancy

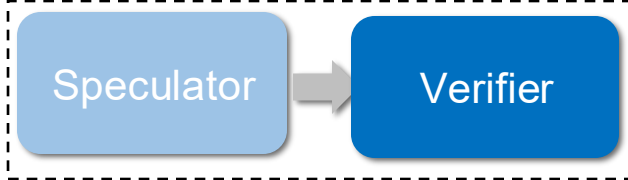
Token Sequences

Attention(
 [machine] learning algorithm
 [machine] learning system design
 [machine] translation model)

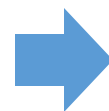
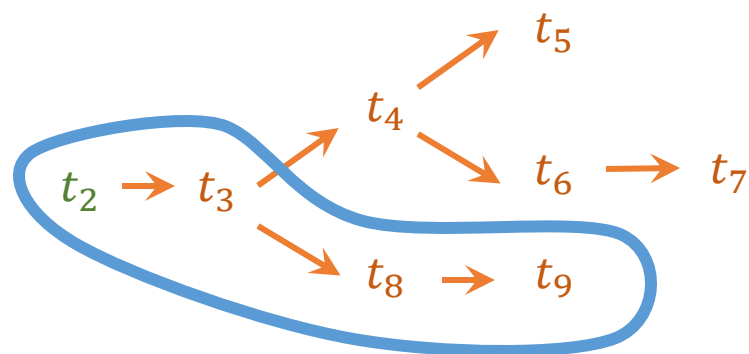
= Tree-Attention(



Token Tree



Tree-based Parallel Decoding



Topology-aware
causal mask

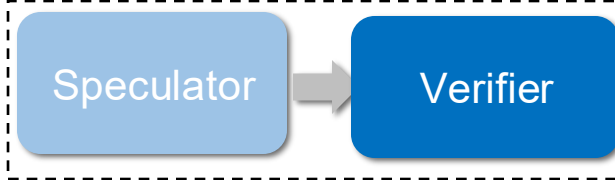
KV-cache

t ₂	t ₃	t ₄	t ₅	t ₆	t ₇	t ₈	t ₉	...
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	-----

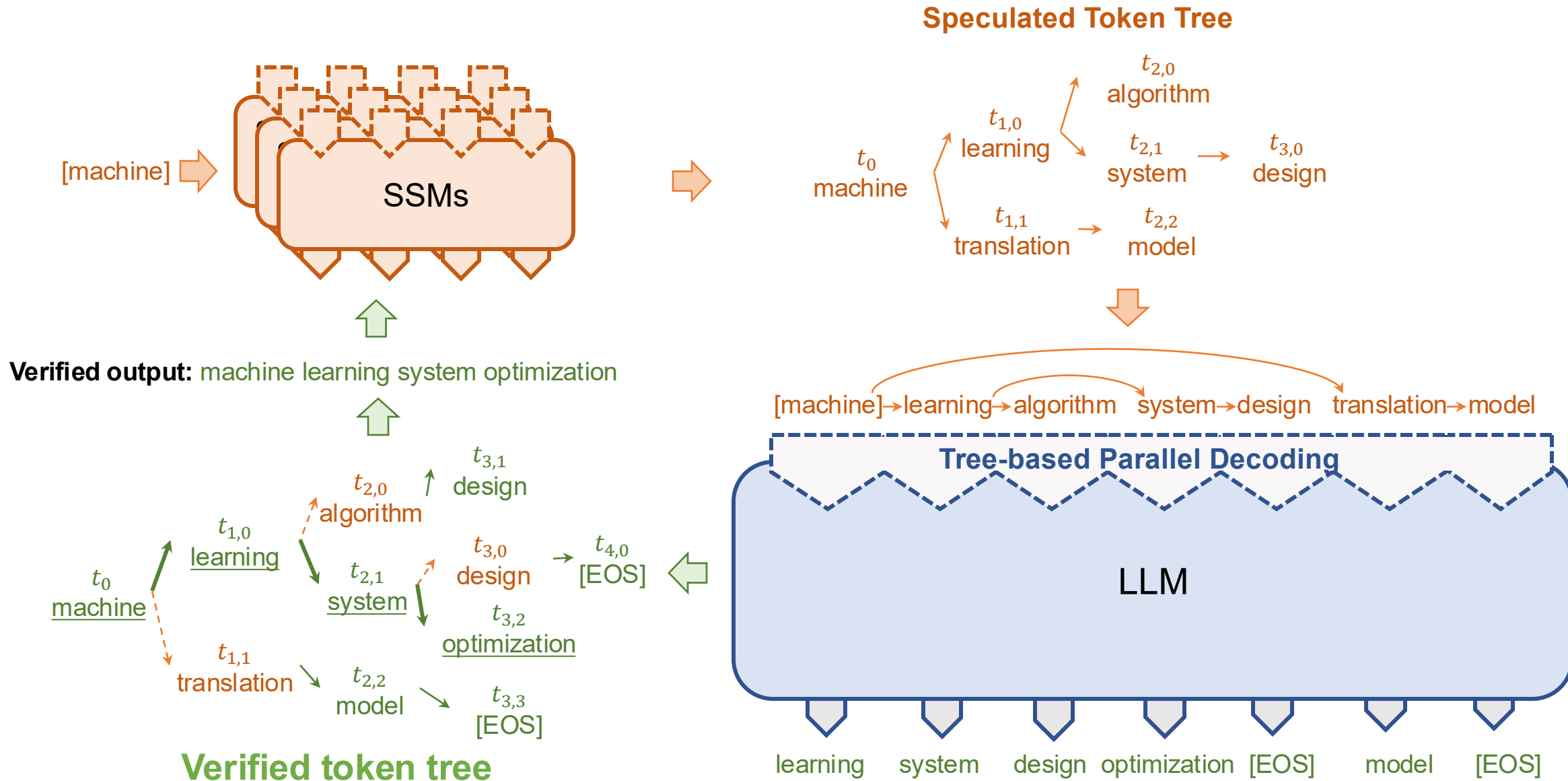
	t ₂	t ₃	t ₄	t ₅	t ₆	t ₇	t ₈	t ₉
t ₂	✓	✓	✓	✓	✓	✓	✓	✓
t ₃		✓	✓	✓	✓	✓	✓	✓
t ₄			✓	✓	✓	✓		
t ₅				✓				
t ₆					✓	✓		
t ₇						✓		
t ₈							✓	✓
t ₉								✓

Key optimizations:

- A DFS-based approach to linearizing a token tree
- Tree topology-aware causal mask
- Decoding all tokens in a single GPU kernel



Verification Workflow



Stochastic Decoding

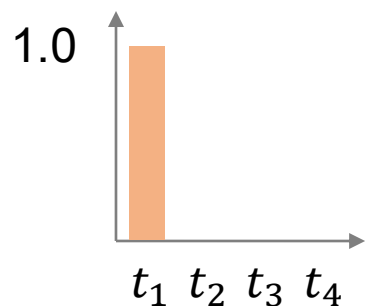
- **Challenge:** verifying stochastic equivalence

$$P_{\text{IncrDecode}}(\cdot | x_{<i}, \text{LLM}) = P_{\text{SpecInfer}}(\cdot | x_{<i}, \text{LLM}, \{\text{SSM}_i\})$$

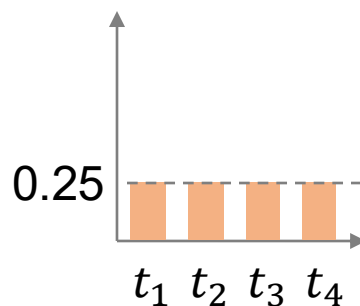
- A strawman approach: **naïve sampling**
 - Use LLM to sample $x_i \sim P_{\text{IncrDecode}}(\cdot | x_{<i}, \text{LLM})$
 - Verify if x_i is in the token tree

Naïve Sampling can be Suboptimal

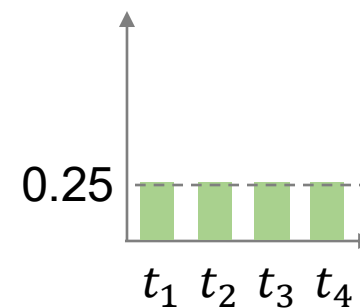
- Assume one LLM, two SSMs, and four possible tokens: t_1, t_2, t_3, t_4



SSM 1: $P(\cdot | x_{<i}, \text{SSM}_1)$



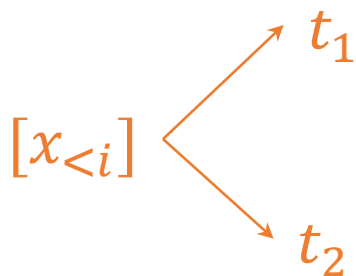
SSM 2: $P(\cdot | x_{<i}, \text{SSM}_2)$



LLM: $P(\cdot | x_{<i}, \text{LLM})$

Naïve sampling's verification prob. = **50%**

But we can do better by directly accepting SSM 2;
verification prob. = **100%**



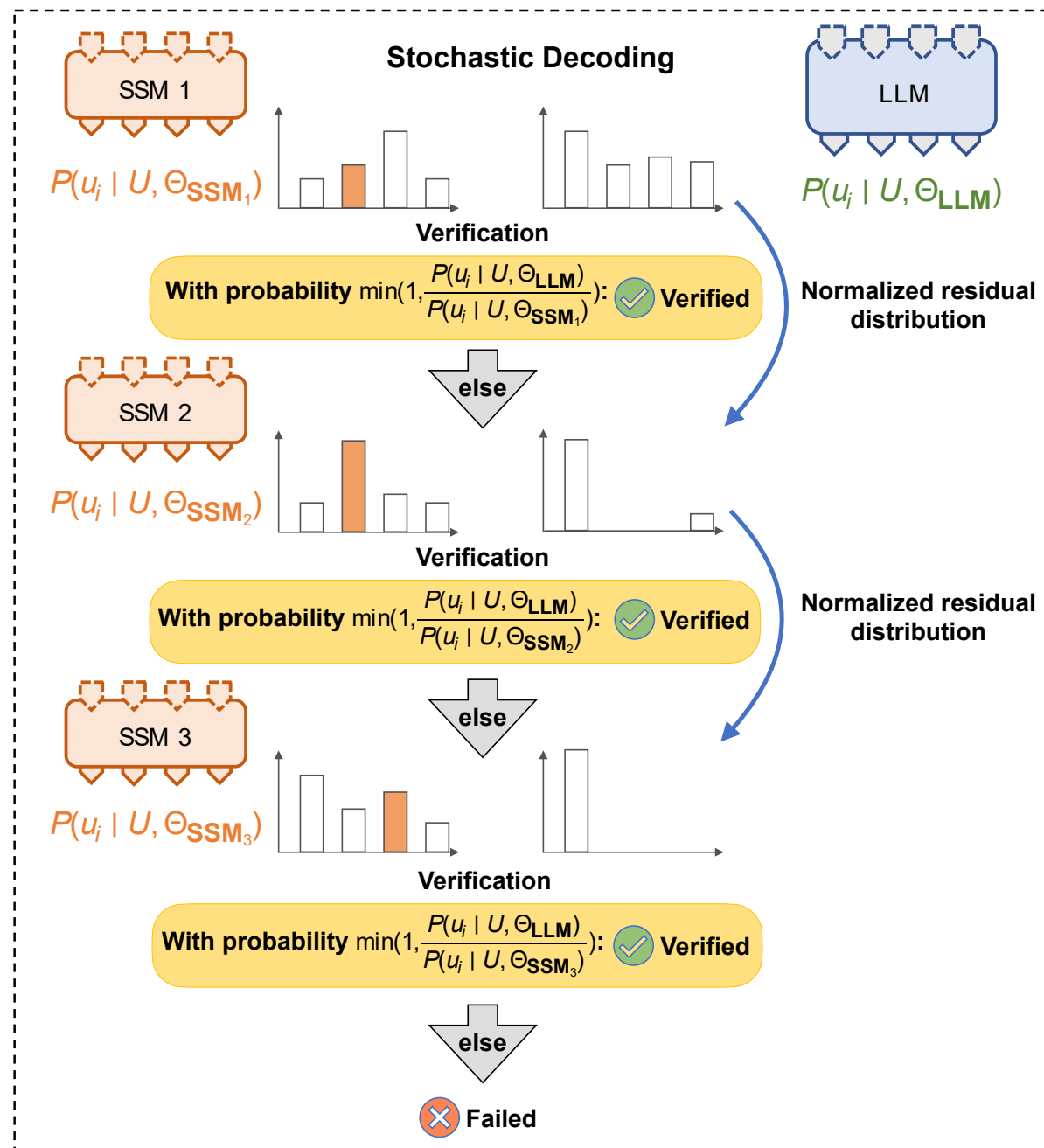
Token Tree

Key issue: naïve sampling ignores correlation
between $P(\cdot | x_{<i}, \text{SSM})$ and $P(\cdot | x_{<i}, \text{LLM})$

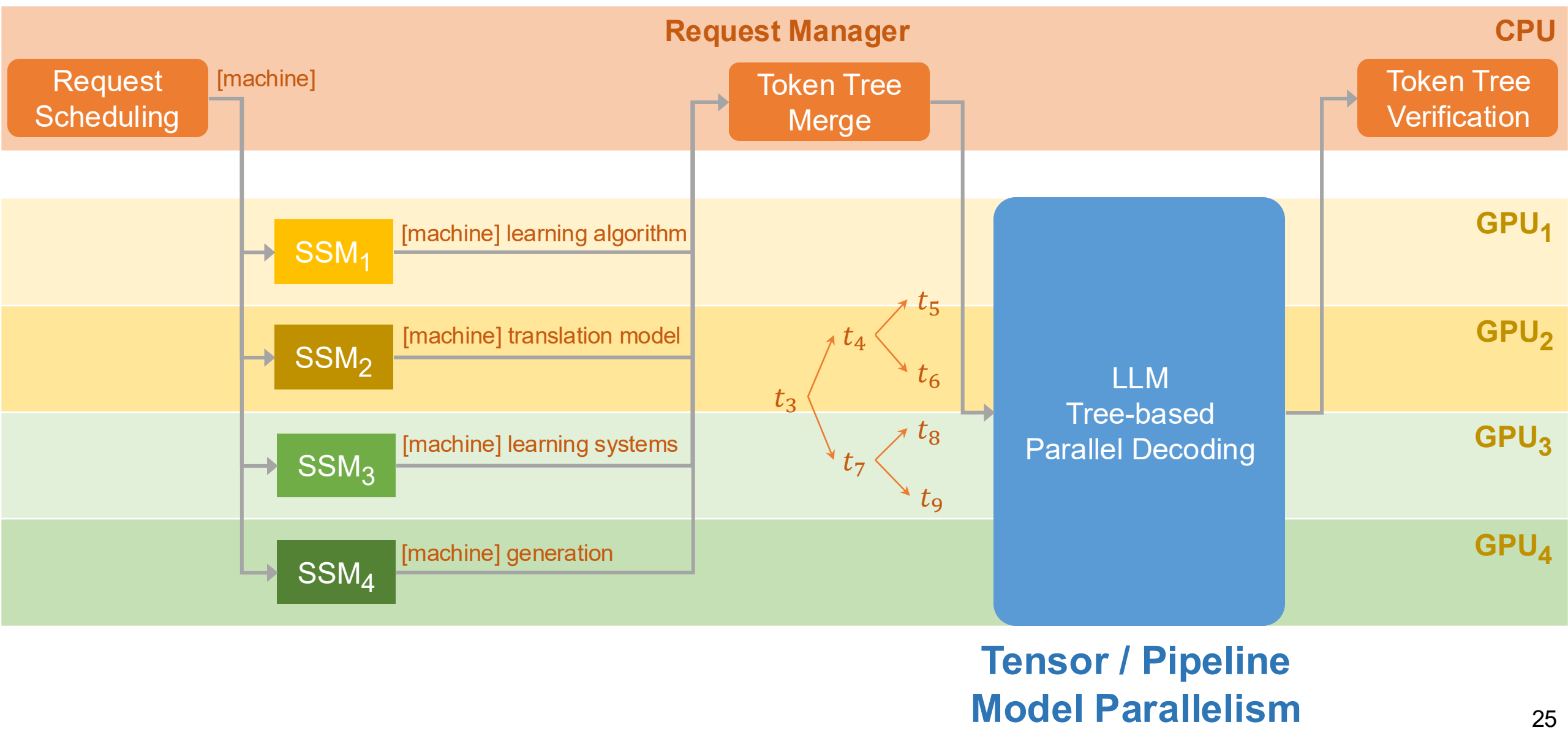
Speculative Sampling

1. Sample a token $x \sim P(u_i | U, \Theta_{SSM})$ using SSM
2. If $P(x | U, \Theta_{SSM}) \leq P(x | U, \Theta_{LLM})$, directly accept x
3. If $P(x | U, \Theta_{SSM}) > P(x | U, \Theta_{LLM})$, accept x with prob. $\frac{P(x | U, \Theta_{LLM})}{P(x | U, \Theta_{SSM})}$
4. If reject x , normalize residual distribution

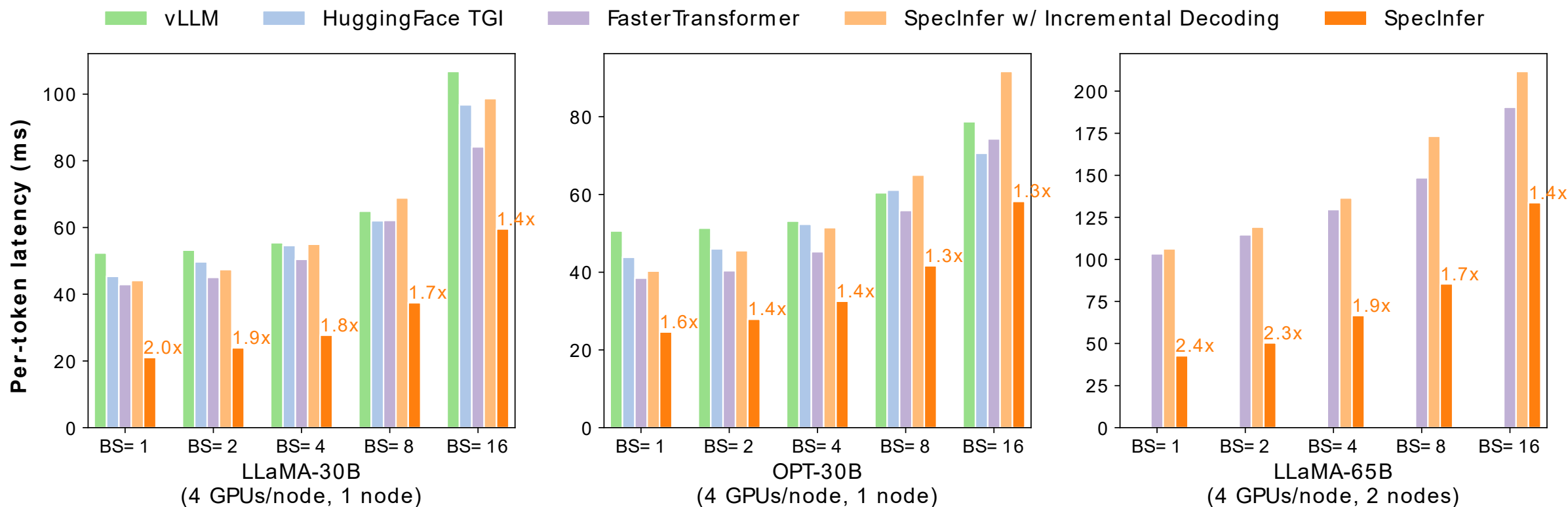
$$P'(x | U, \Theta_{LLM}) = \text{norm}(\max(0, P(x | U, \Theta_{LLM}) - P(x | U, \Theta_{SSM})))$$



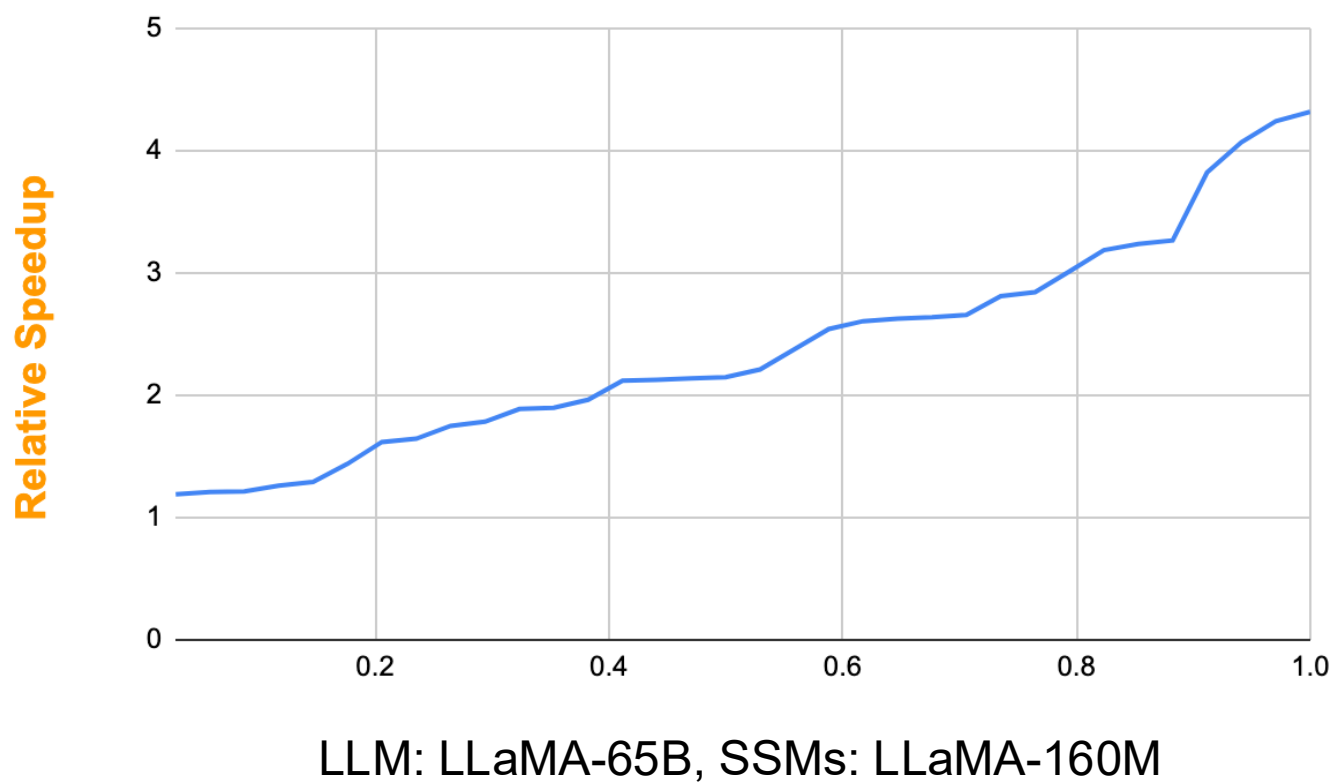
Distributed LLM Serving



SpecInfer Accelerates LLM Inference by 1.3-2.4x

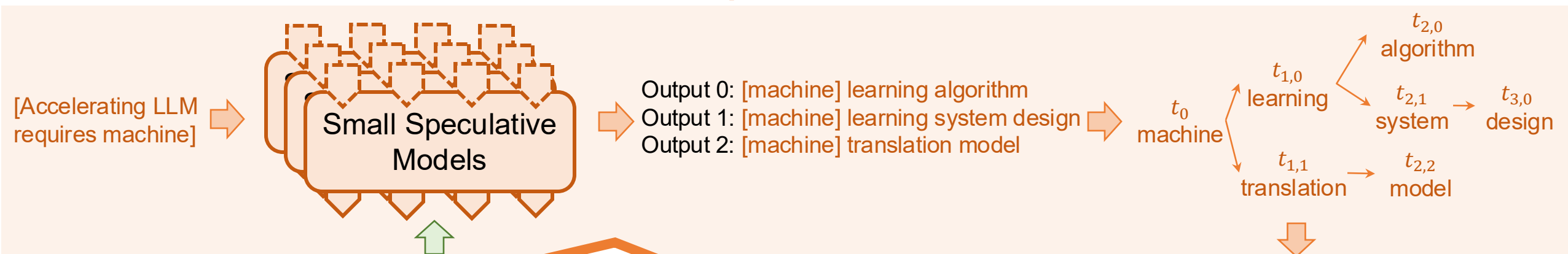


SpecInfer can Consistently Accelerate LLM Inference



Open Research Questions

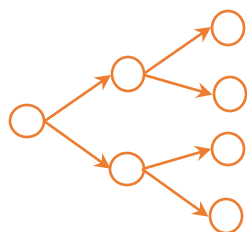
Speculator



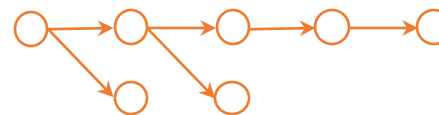
Verified output: machine

Balanced v.s. unbalanced trees?

t_0 machine
 $t_{1,0}$ learning
 $t_{1,1}$ translation

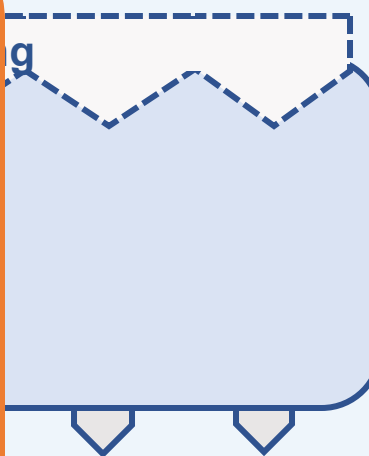


Balanced trees:
lower speculative overhead



Unbalanced trees:
higher verification prob.

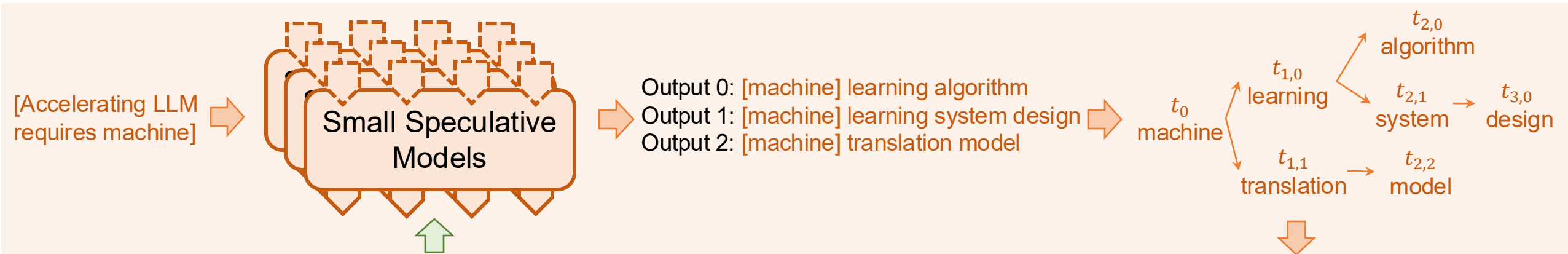
translation → model



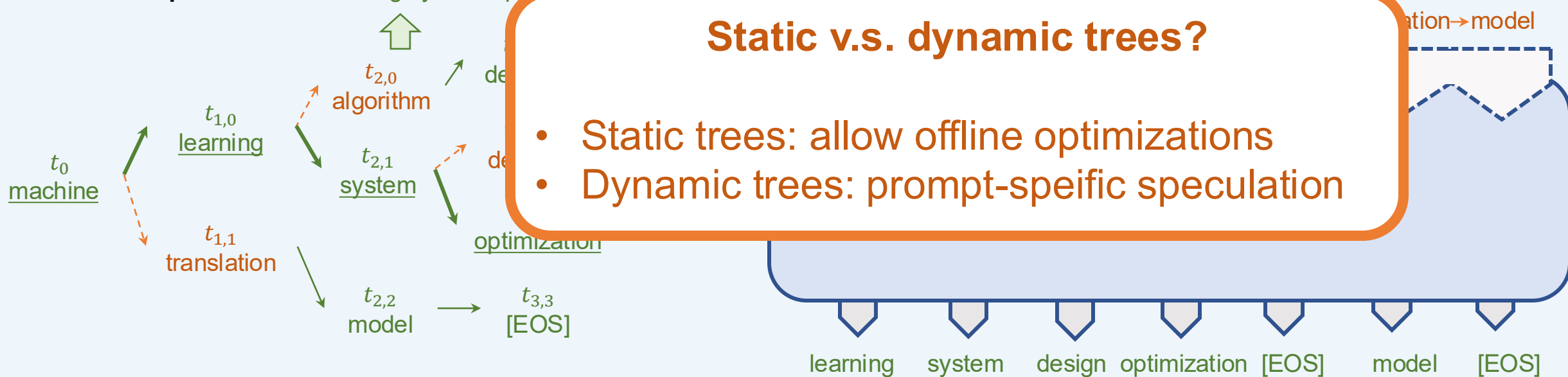
model [EOS]

Open Research Questions

Speculator



Verified output: machine learning system optimization



Static v.s. dynamic trees?

- Static trees: allow offline optimizations
- Dynamic trees: prompt-specific speculation

Verifier

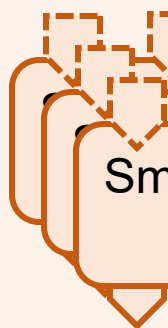
Open Research Questions

Speculator

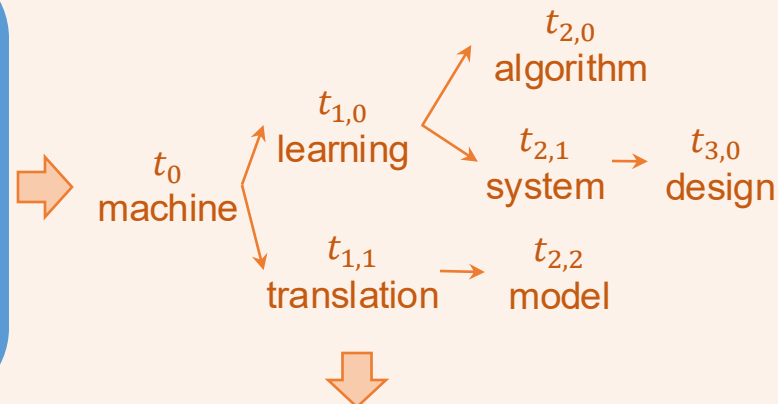
How to verify stochastic decoding?

- Multi-step speculative sampling
- Naïve sampling
- Greedy sampling
- Mixture?

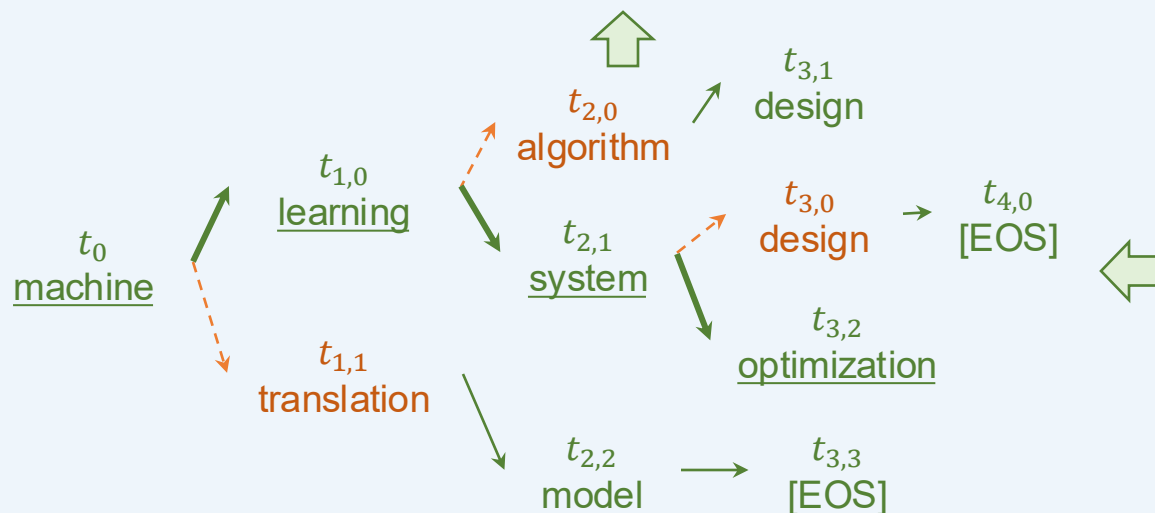
[Accelerating LLM requires machine]



Sm



Verified output: machine learning system optimization



[machine] → learning → algorithm → system → design → translation → model

Tree-based Parallel Decoding

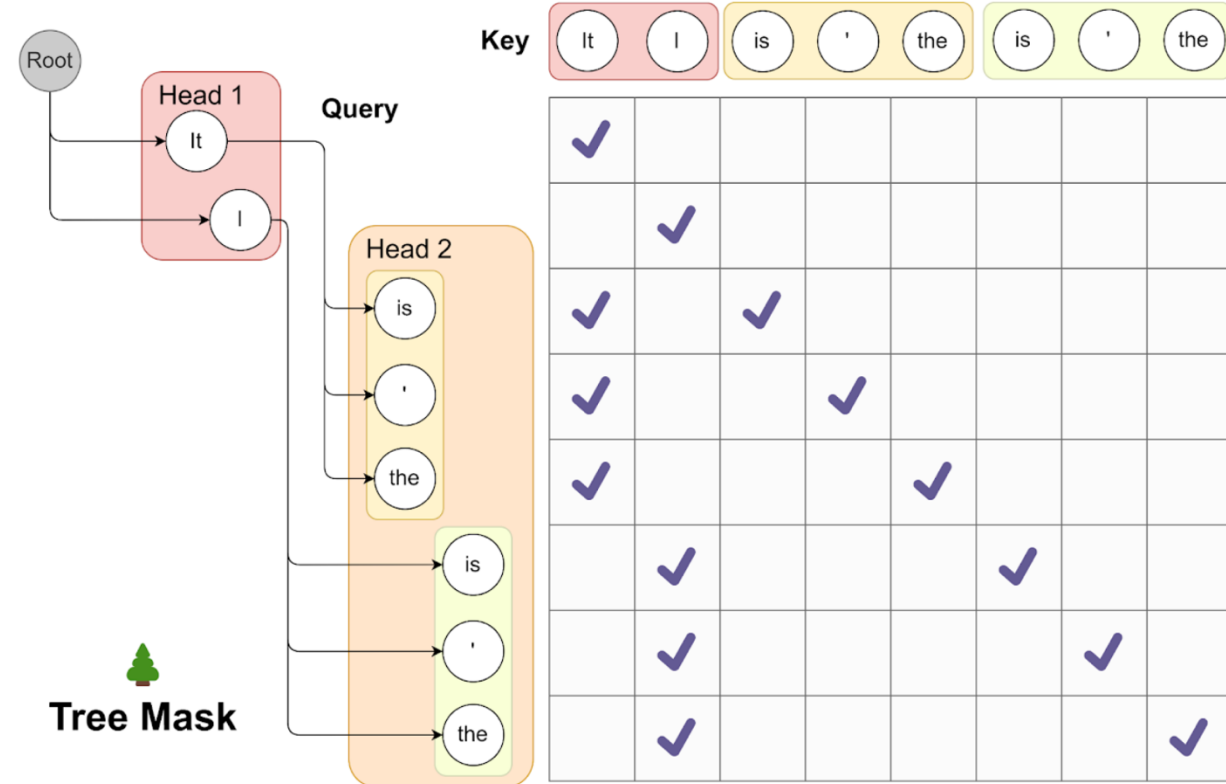
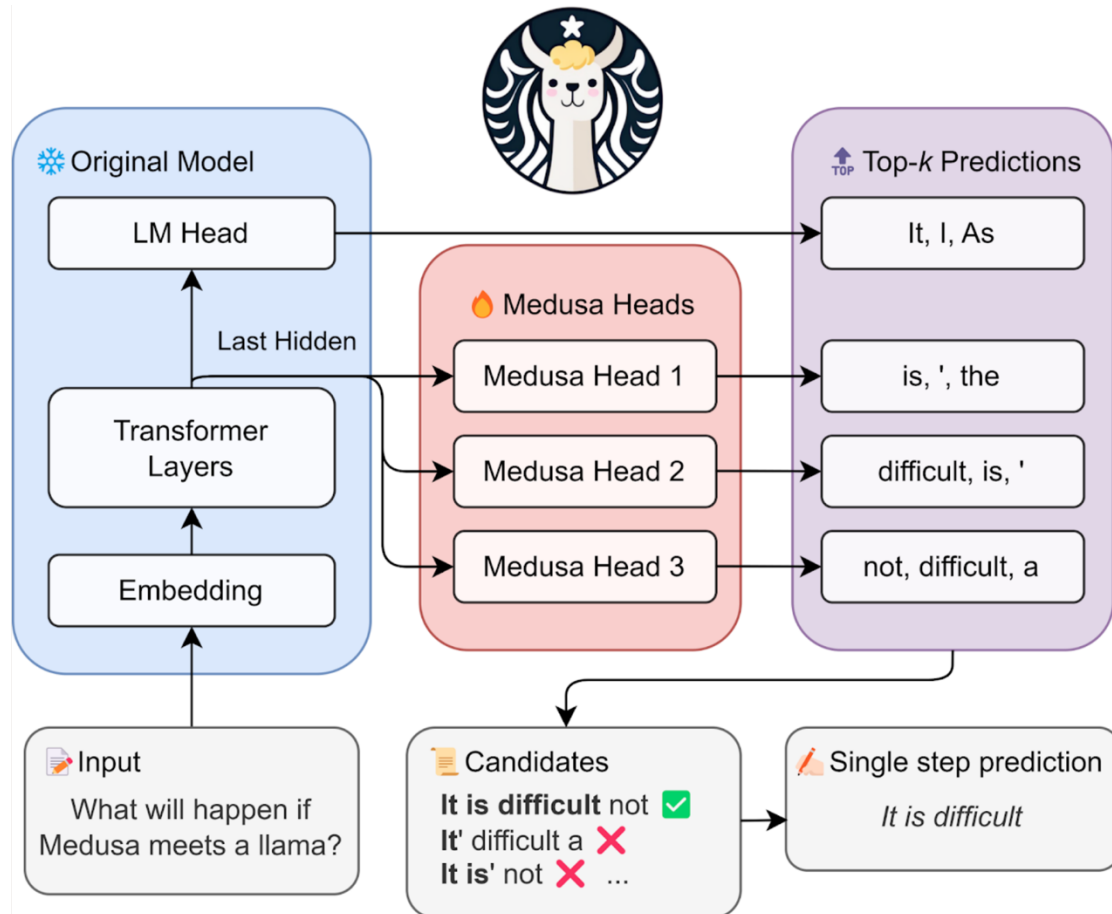
LLM

learning system design optimization [EOS] model [EOS]

Verifier

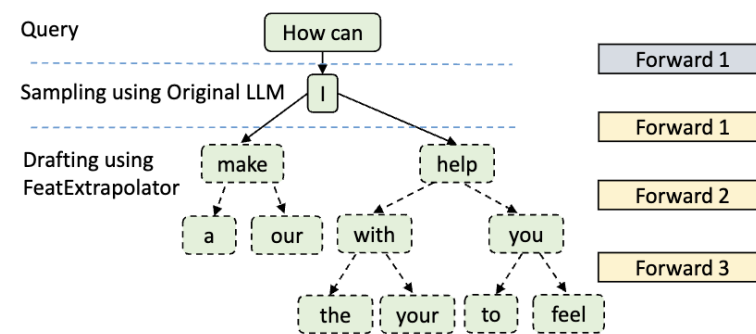
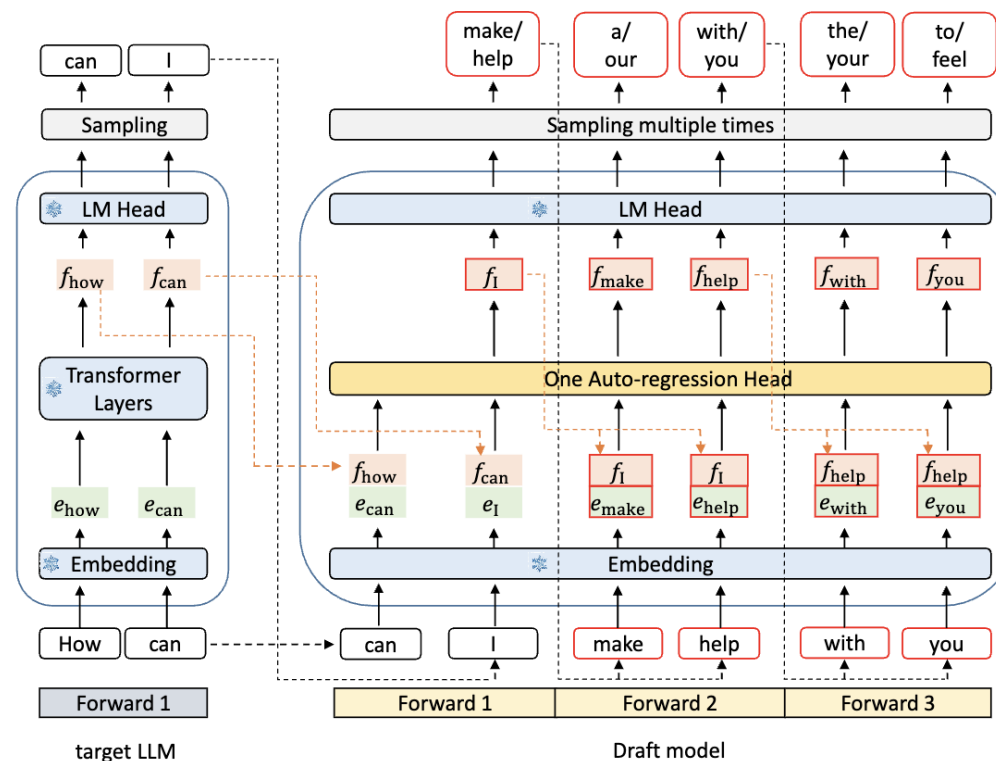
Medusa: Speculative Decoding with Multiple Decoding Heads

- Introduce trainable attention heads to predict future tokens



Eagle: Dynamic Speculative Token Trees

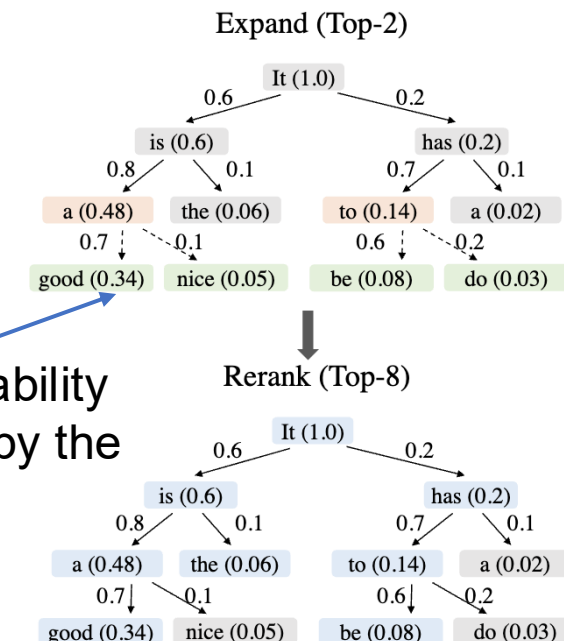
- Draft model reuses embedding and LM head layers of the LLM, and introduces a trainable **attention layer**
- **Expand phase**: draft model selects K nodes to decode in each iteration



Eagle: Dynamic Speculative Token Trees

- Draft model reuses embedding and LM head layers of the LLM, and introduces a trainable **attention layer**
- **Expand phase**: draft model selects K nodes to decode in each iteration
- **Rerank phase**: select N nodes for verification based on accumulated likelihood

Accumulated probability to select that path by the draft model



Flatten to 1D

It	is	has	a	the	to	good	be
----	----	-----	---	-----	----	------	----

Attention mask

	It	is	has	a	the	to	good	be
It	✓							
is	✓	✓						
has	✓		✓					
a	✓	✓		✓				
the	✓	✓			✓			
to	✓		✓			✓		
good	✓	✓		✓			✓	
be	✓		✓			✓		✓

Today's Lecture: Speculative Decoding

- Model-based speculative decoding
 - Using a separate draft model to predict LLM's output
- **Model-free speculative decoding**
 - Using previously generated tokens to predict future tokens

Prompt Lookup Decoding

- **Prompt:** What is the capital of South Korea?

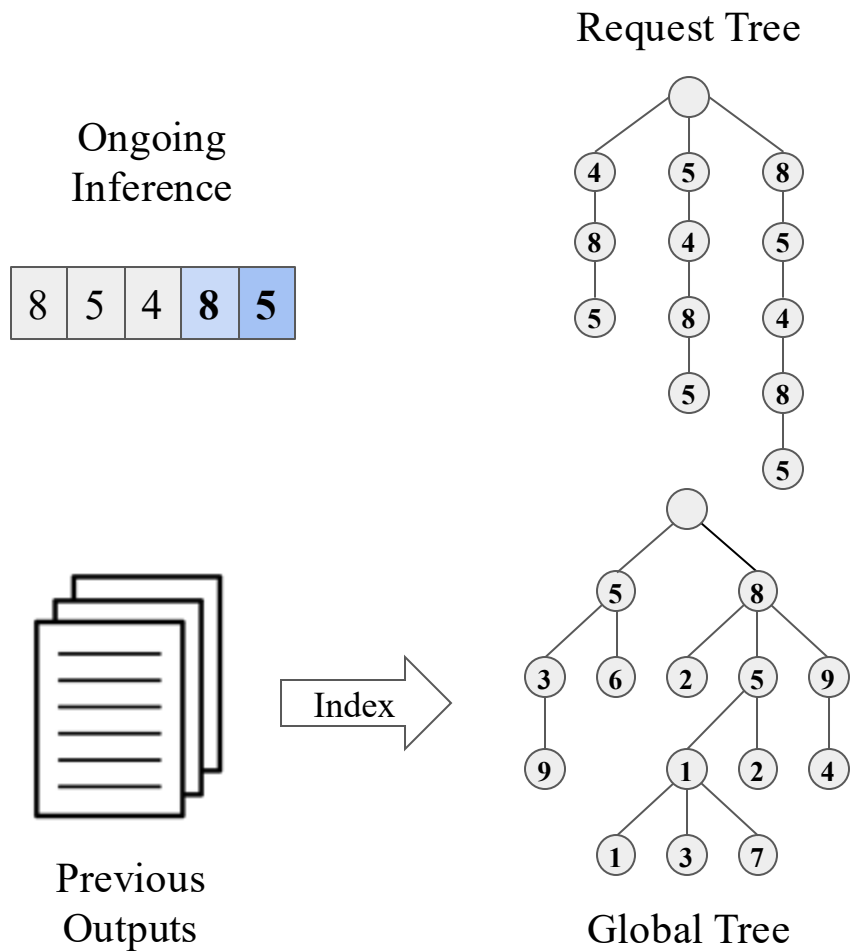
2-gram	3 speculate tokens
What is	the capital of
is the	capital of South
the capital	of South Korea
capital of	South Korea ?

During generation:

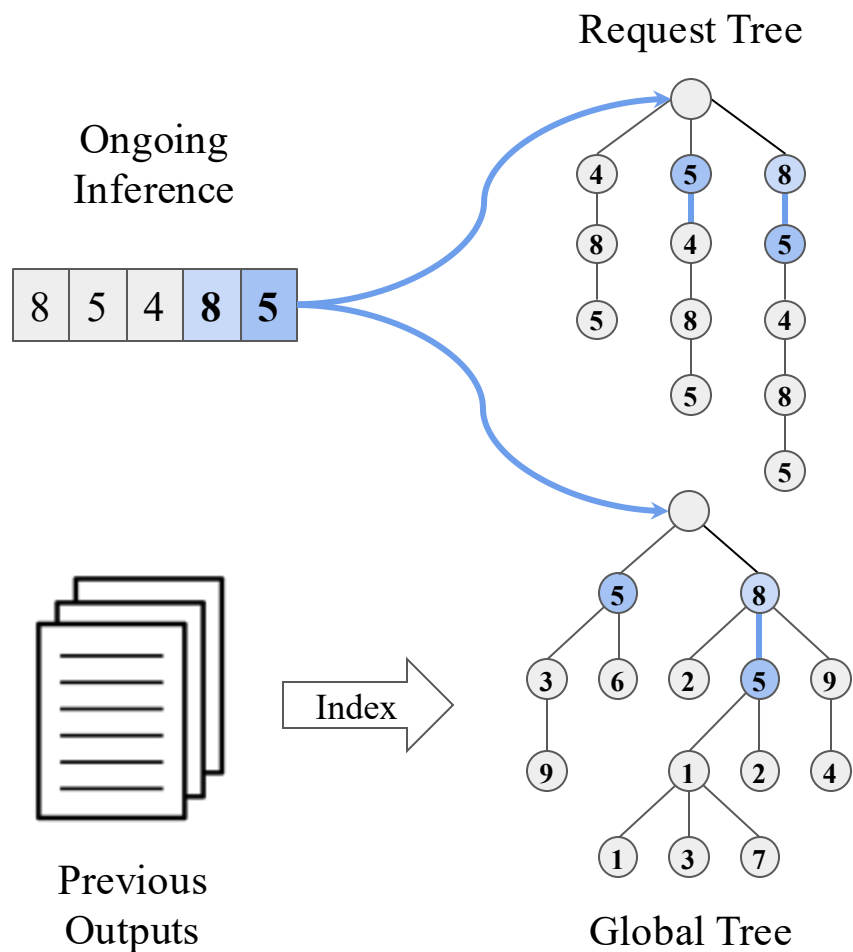
the capital of South Korea



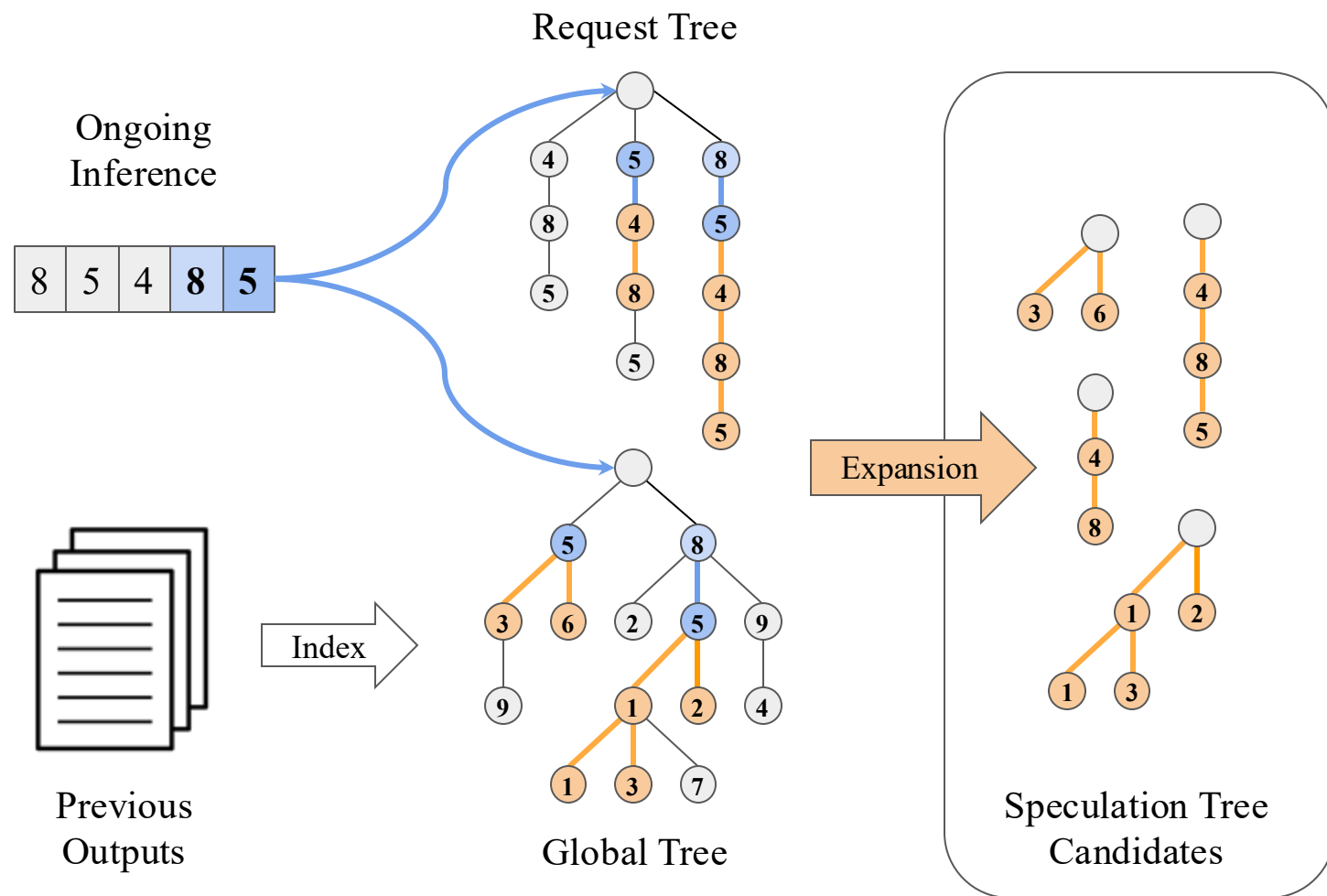
SuffixDecoding: Two-Tier Suffix Tree Speculation



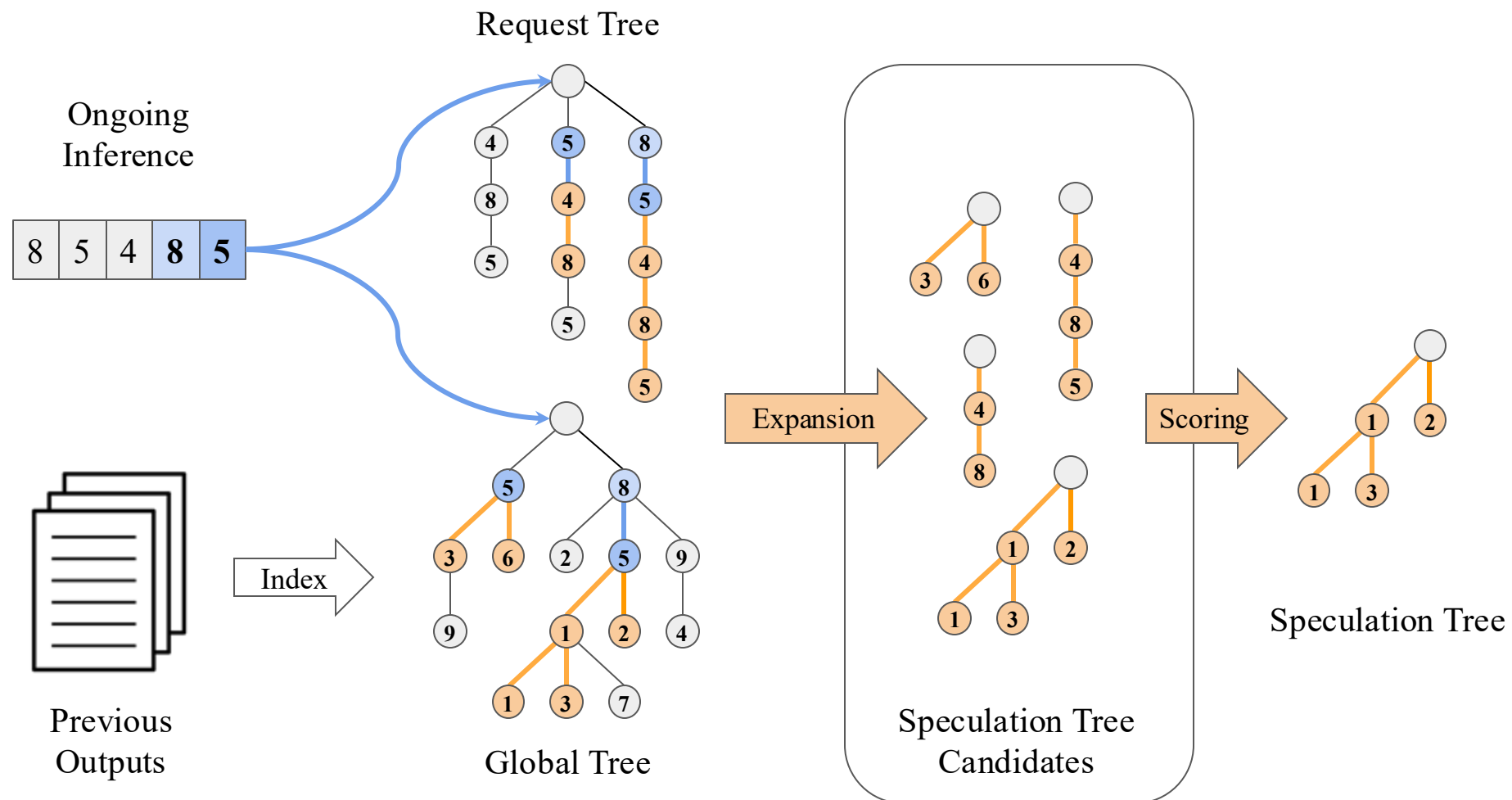
SuffixDecoding: Two-Tier Suffix Tree Speculation



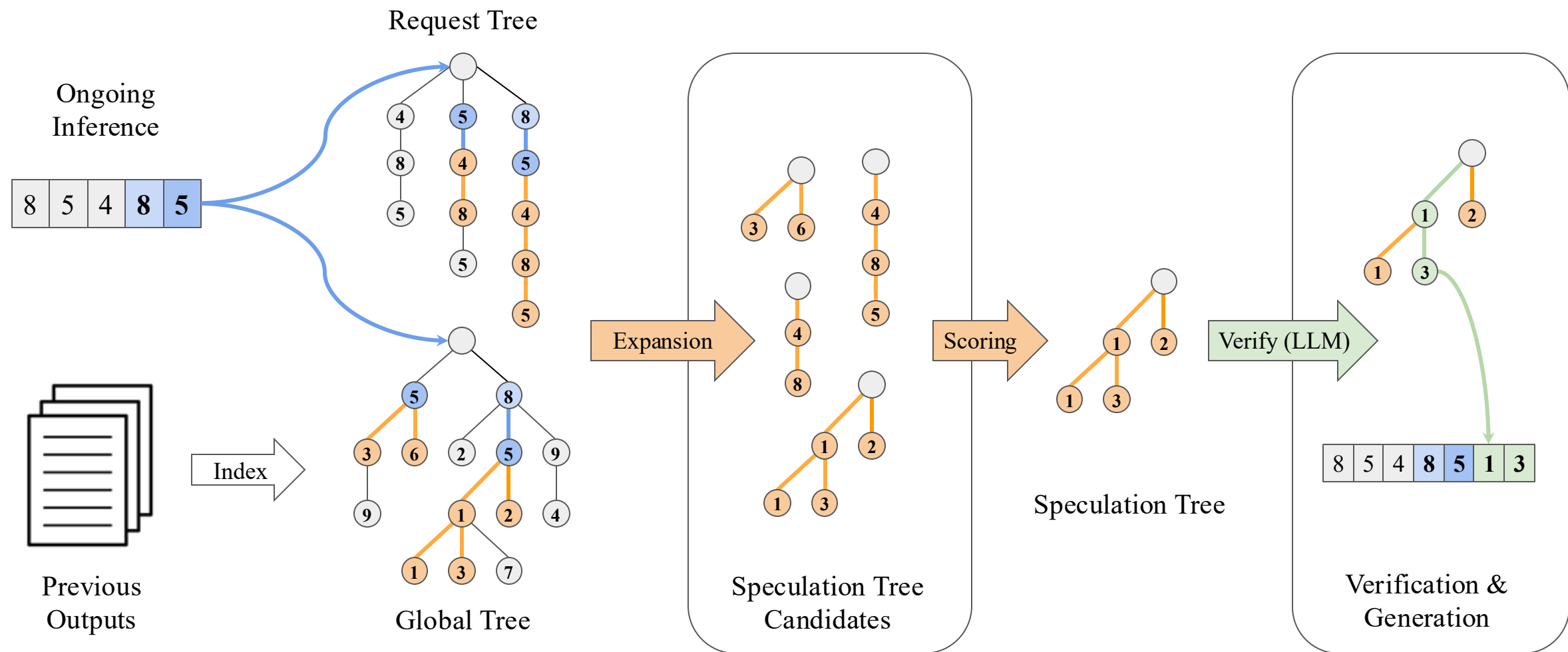
SuffixDecoding: Two-Tier Suffix Tree Speculation



SuffixDecoding: Two-Tier Suffix Tree Speculation

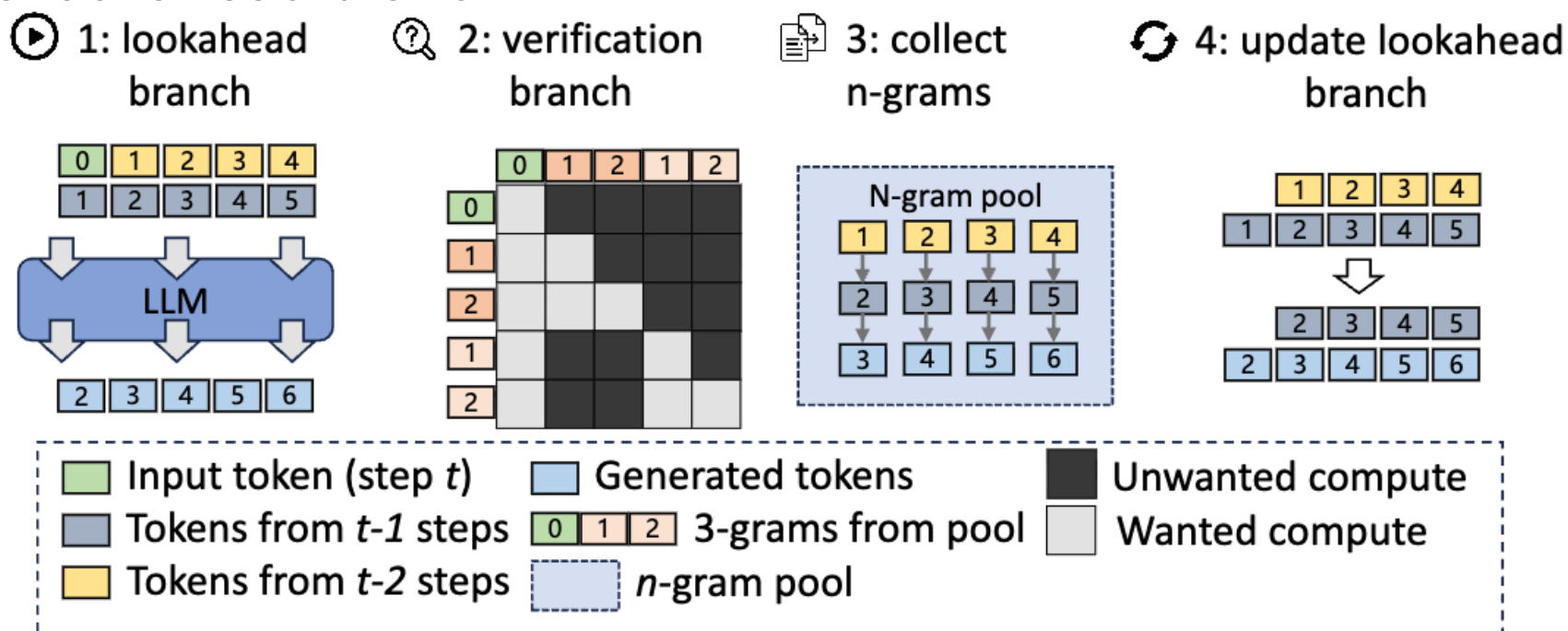


SuffixDecoding: Two-Tier Suffix Tree Speculation



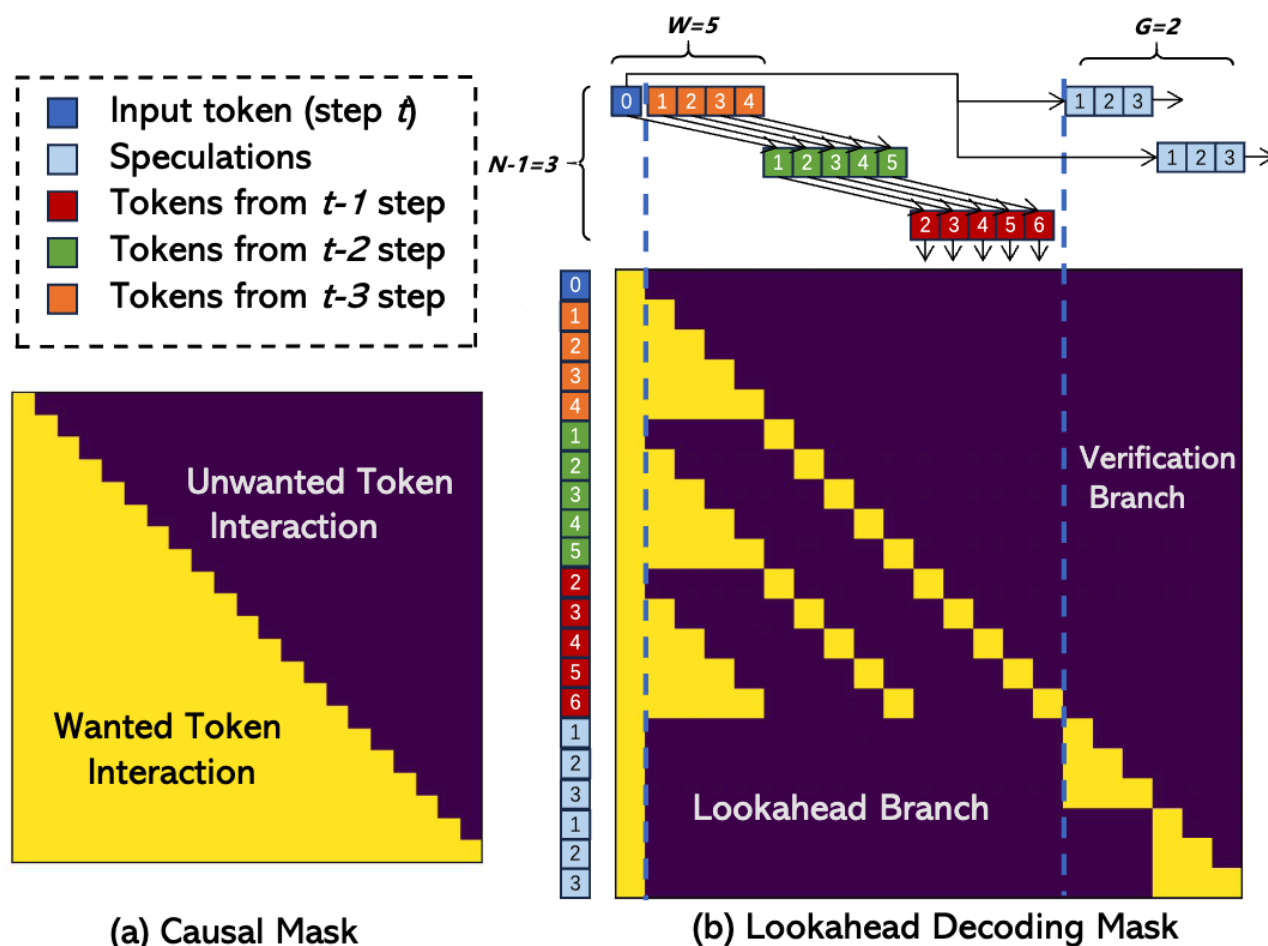
Lookahead Decoding

1. Lookahead branch: generate one token at each position
2. Verification branch: verify multiple 3-grams (searched from the 3-gram pool)
3. Collect and cache newly generated 3-grams from lookahead branch
4. Update lookahead branch



Lookahead Decoding

- Batch verification and lookahead branches in a single pass



(a) Causal Mask

(b) Lookahead Decoding Mask

Today's Lecture: Speculative Decoding

- Model-based speculative decoding
 - Using a separate draft model to predict LLM's output
- Model-free speculative decoding
 - Using previously generated tokens to predict future tokens