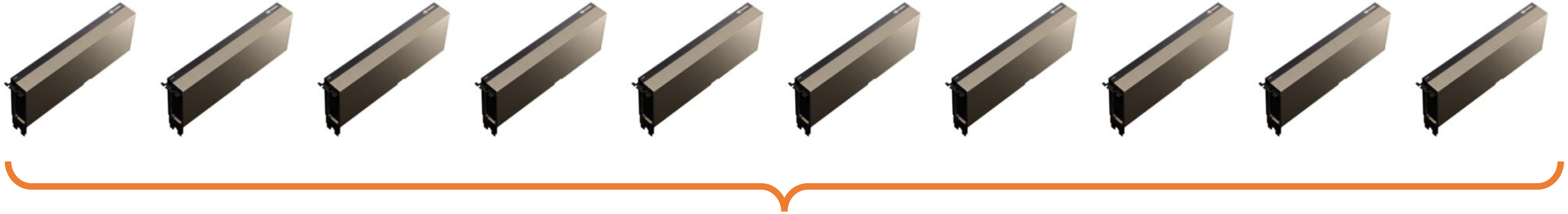
15-442/15-642: Machine Learning Systems

LLMs Serving Techniques

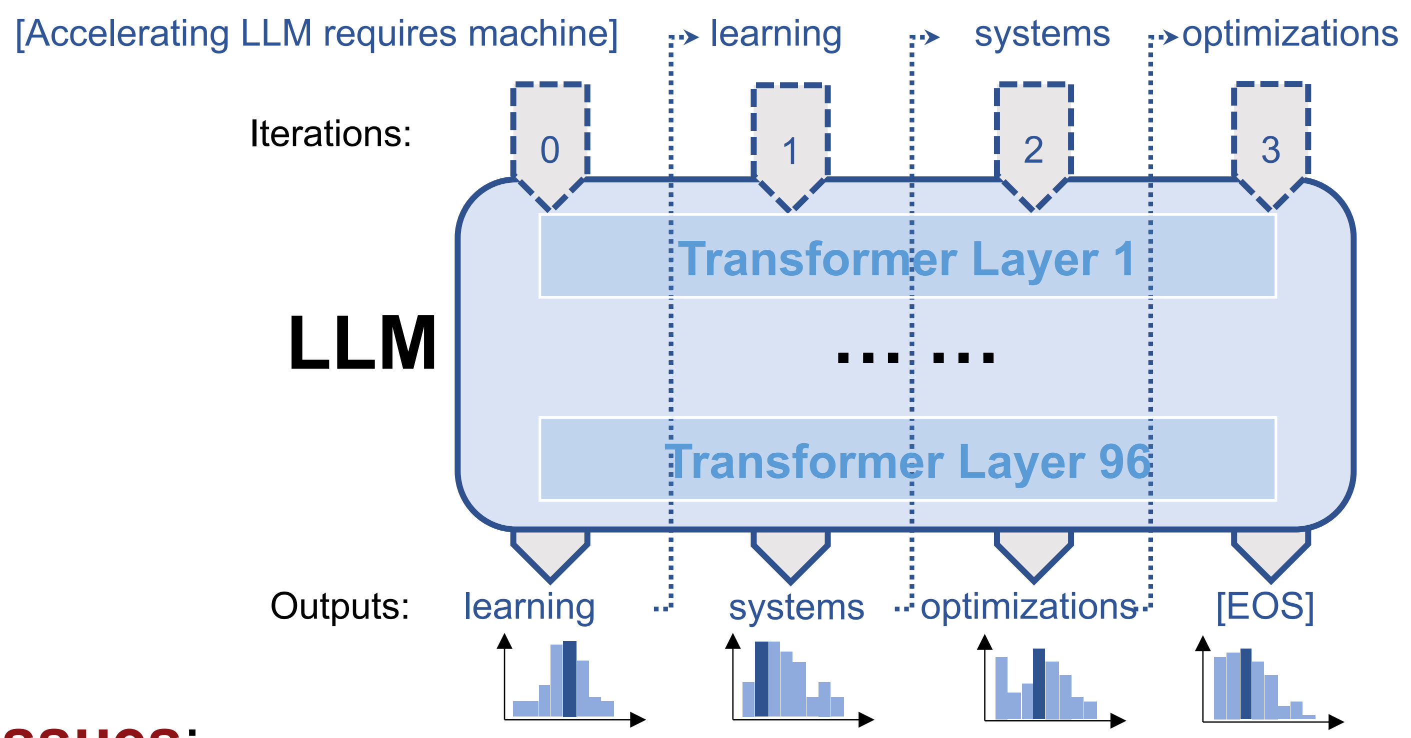
Tianqi Chen and Zhihao Jia
Carnegie Mellon University

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**

Recall: Incremental Decoding



Main issues:

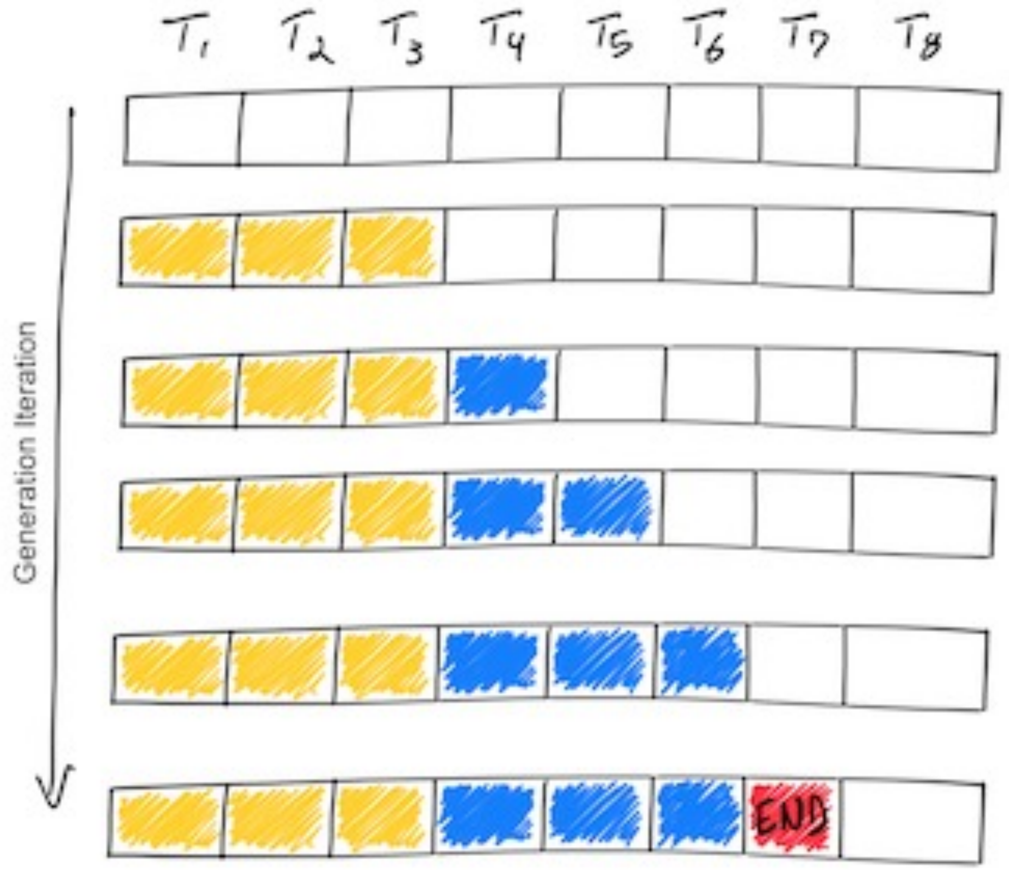
- Limited degree of parallelism → underutilized GPU resources
- Need all parameters to decode a token → bottlenecked by GPU memory access

Outline: LLMs Serving Techniques

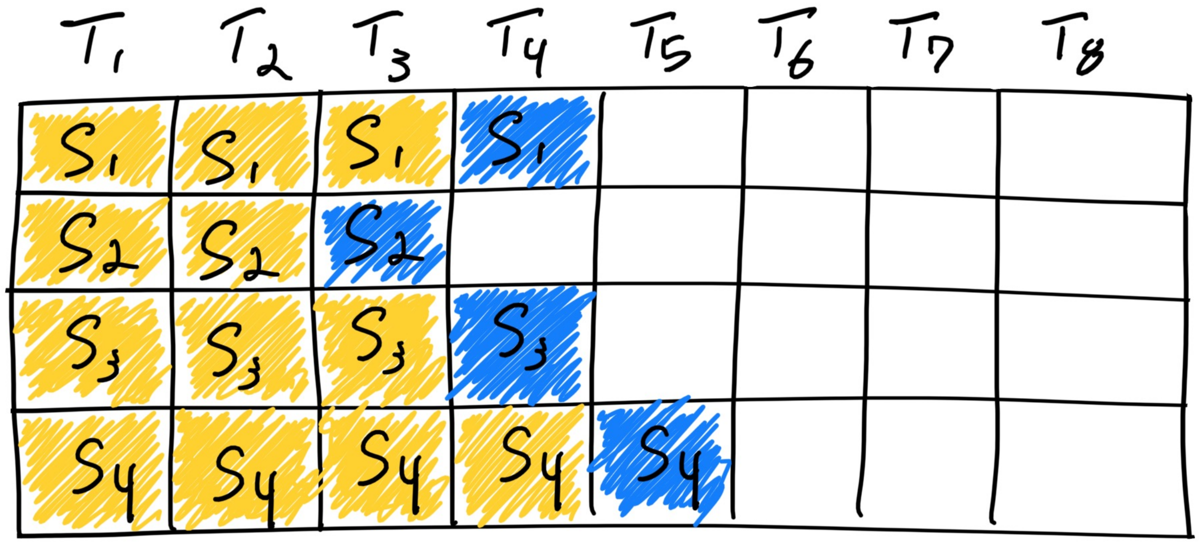
- **Continuous Batching**

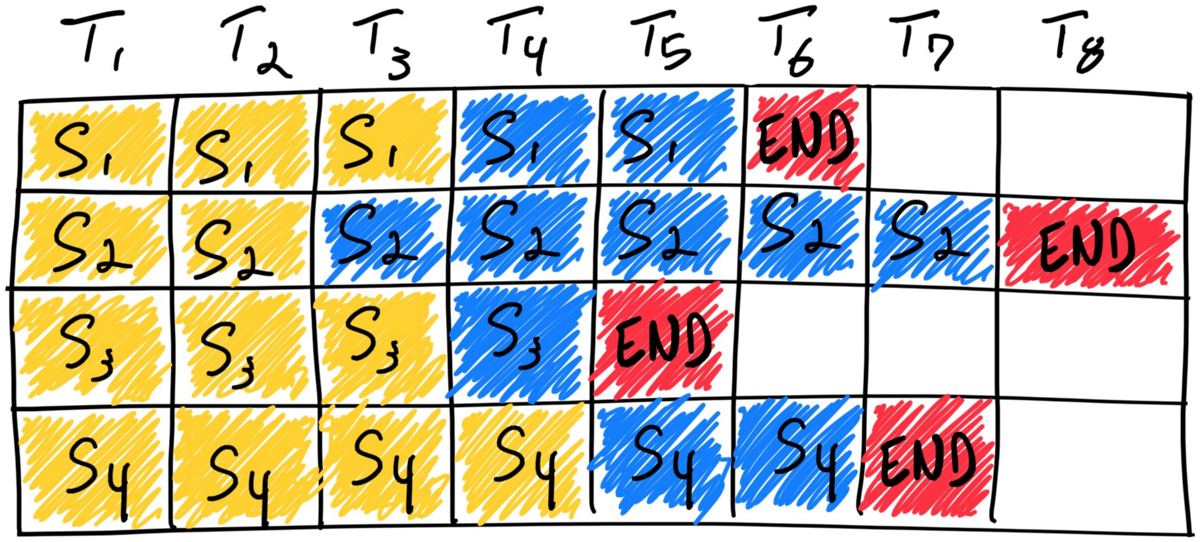
- Speculative Decoding

LLM Decoding Timeline



Batching Requests to Improve GPU Performance

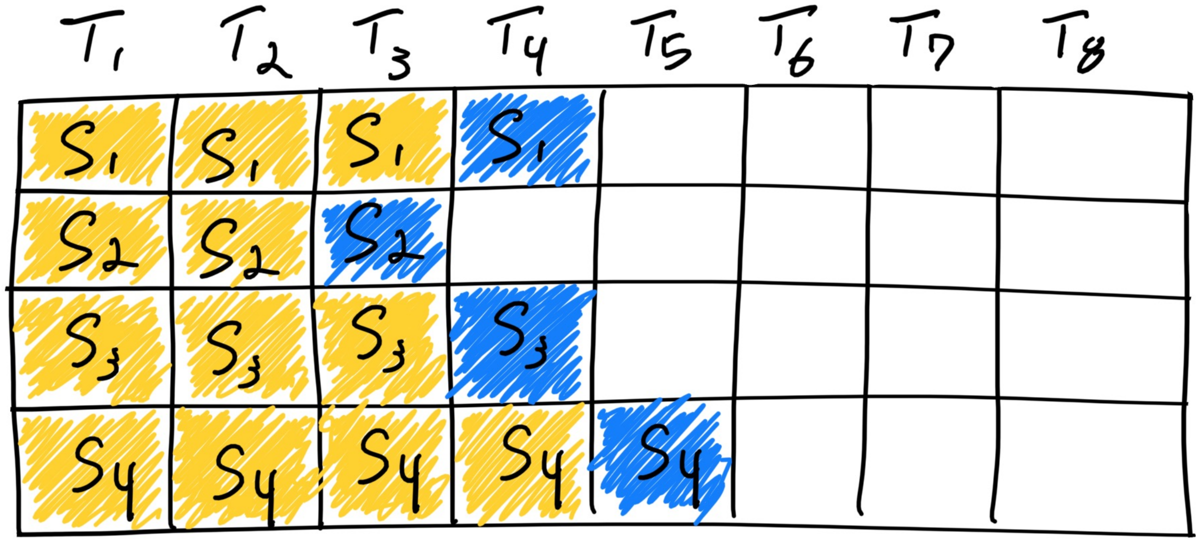


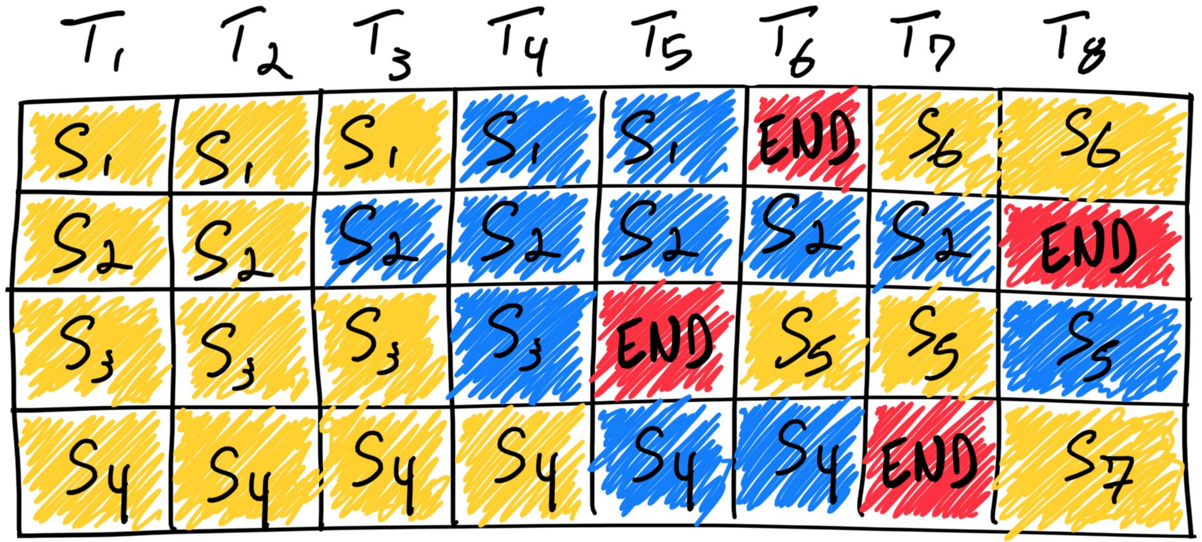


Issues with static batching:

- Requests may complete at different iterations
- Idle GPU cycles
- New requests cannot start immediately

Continuous Batching





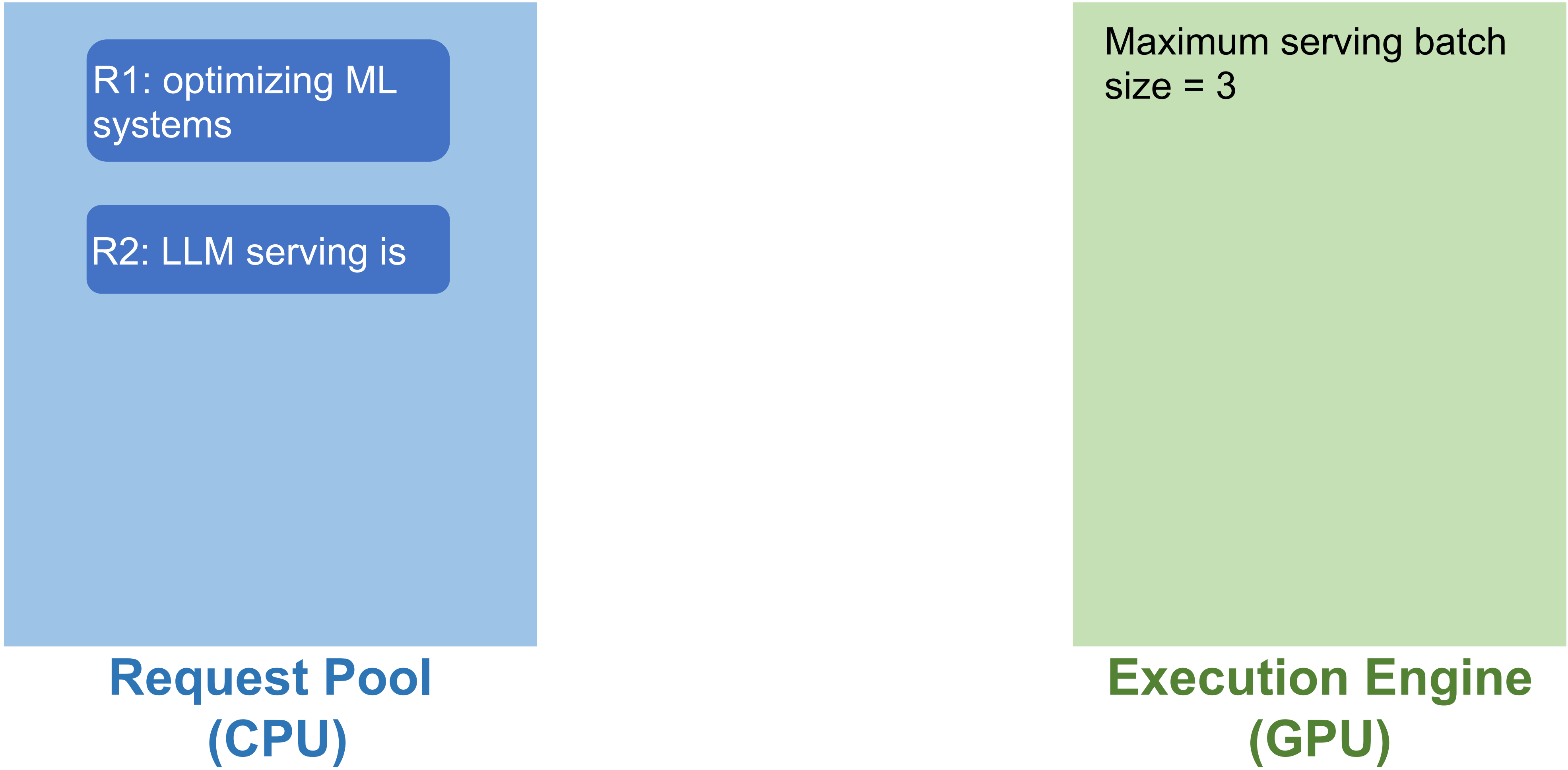
Benefits:

- Higher GPU utilization
- New requests can start immediately

Orca: A Distributed Serving System for Transformer-Based Generative Models. OSDI'22

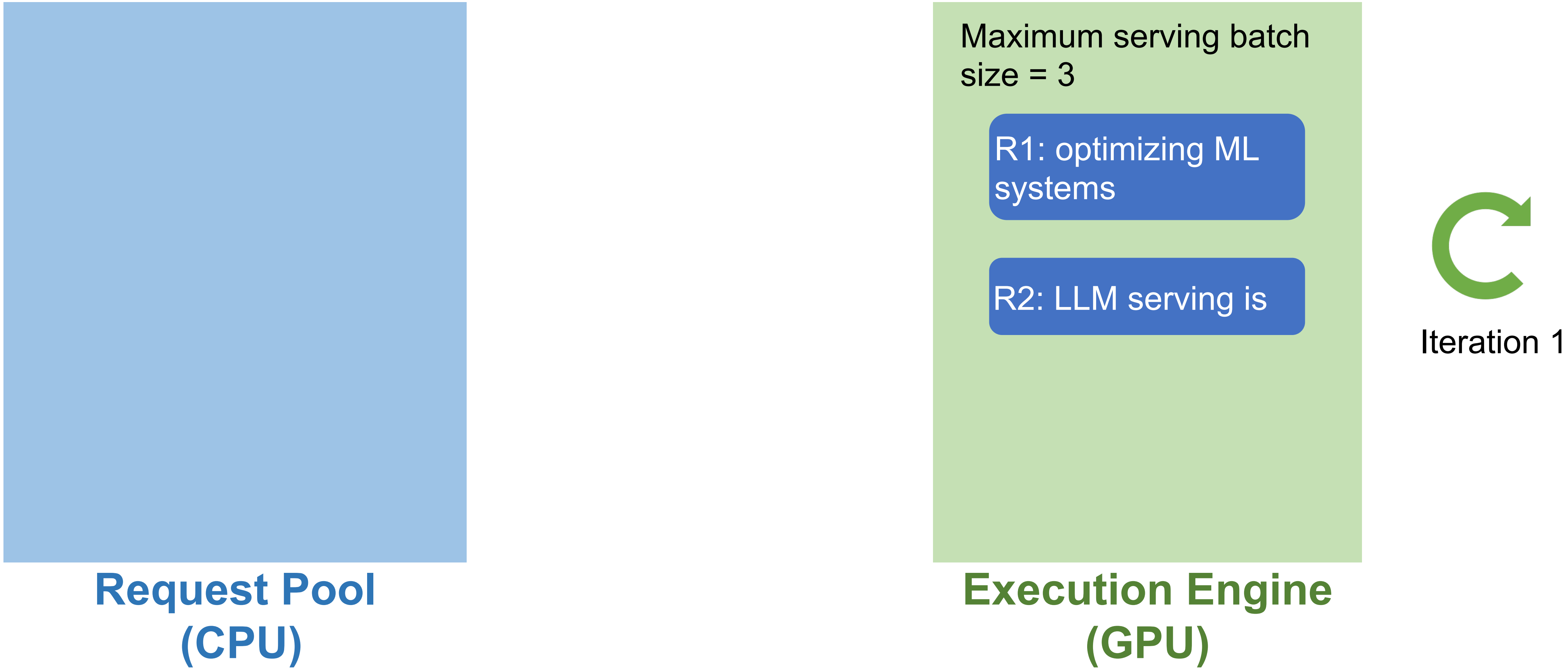
Continuous Batching Step-by-Step

- Receives two new requests R1 and R2



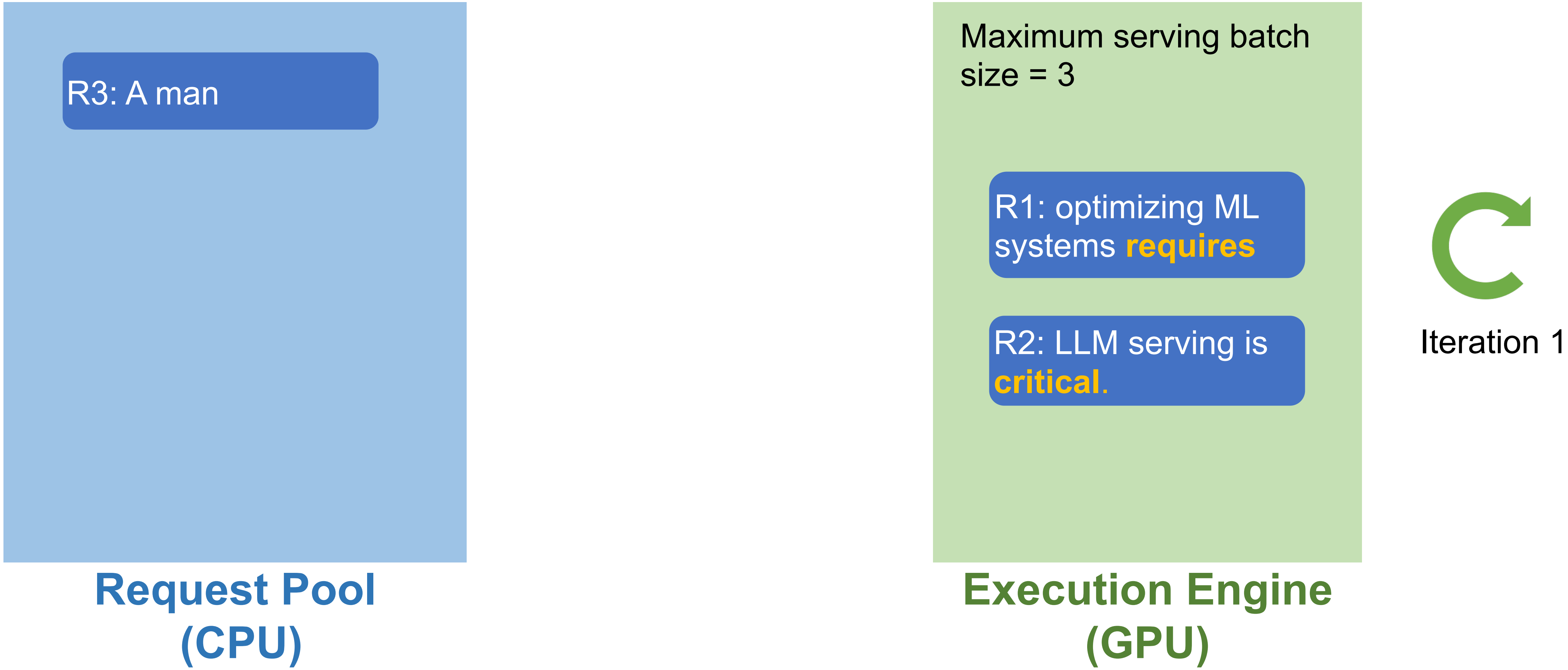
Continuous Batching Step-by-Step

- Iteration 1: decode R1 and R2



Continuous Batching Step-by-Step

- Receive a new request R3; finish decoding R1 and R2



Continuous Batching Step-by-Step

- Iteration 2: decode R1, R2, R3; receive R4, R5; R2 completes

R4: A dog is

R5: How are

Request Pool (CPU)

Maximum serving batch size = 3

R3: A man **is**

R1: optimizing ML systems **requires ML**

R2: LLM serving is **critical**. <EOS>

Execution Engine (GPU)

Iteration 2

Continuous Batching Step-by-Step

- Iteration 3: decode R1, R3, R4

R5: How are

Request Pool (CPU)

Maximum serving batch size = 3

R3: A man **is**

R1: optimizing ML systems **requires ML**

R4: A dog is

Execution Engine (GPU)

Iteration 3

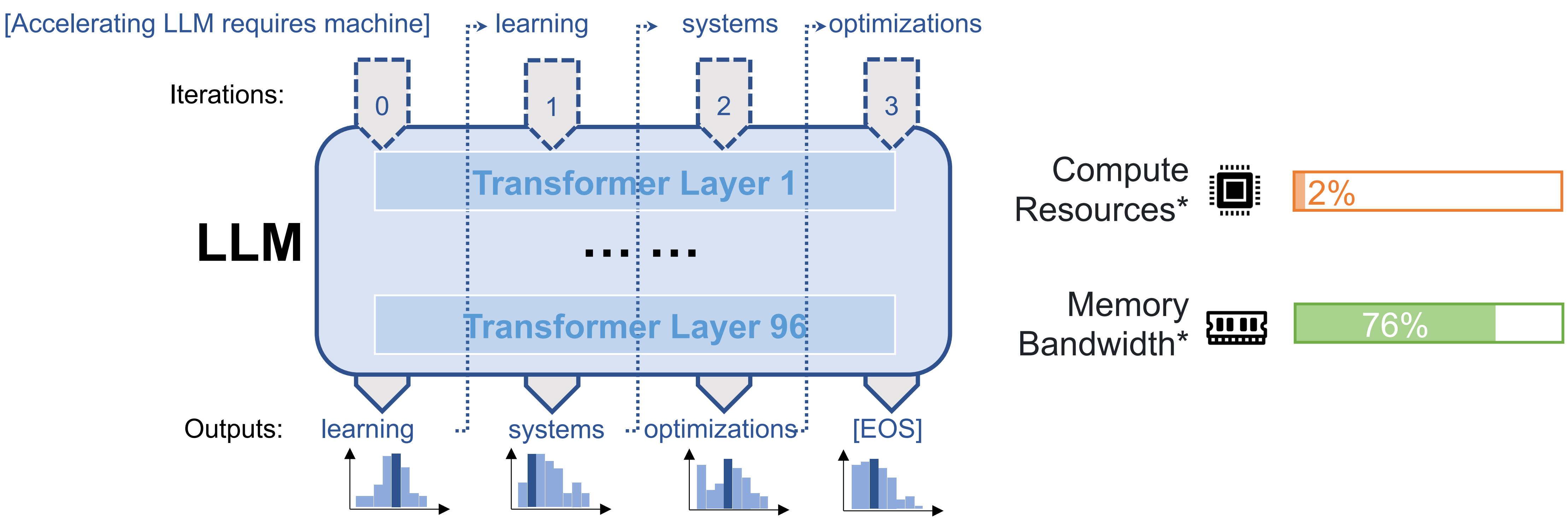
Continuous Batching

- Handle early-finished and late-arrived requests more efficiently
- Higher GPU utilization

Outline: LLMs Serving Techniques

- Continuous Batching
- **Speculative Decoding**

Recall: Incremental Decoding Issues



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

* Measured by serving LLAMA-2-70B on 4 A100 GPUs with 4K sequence length

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

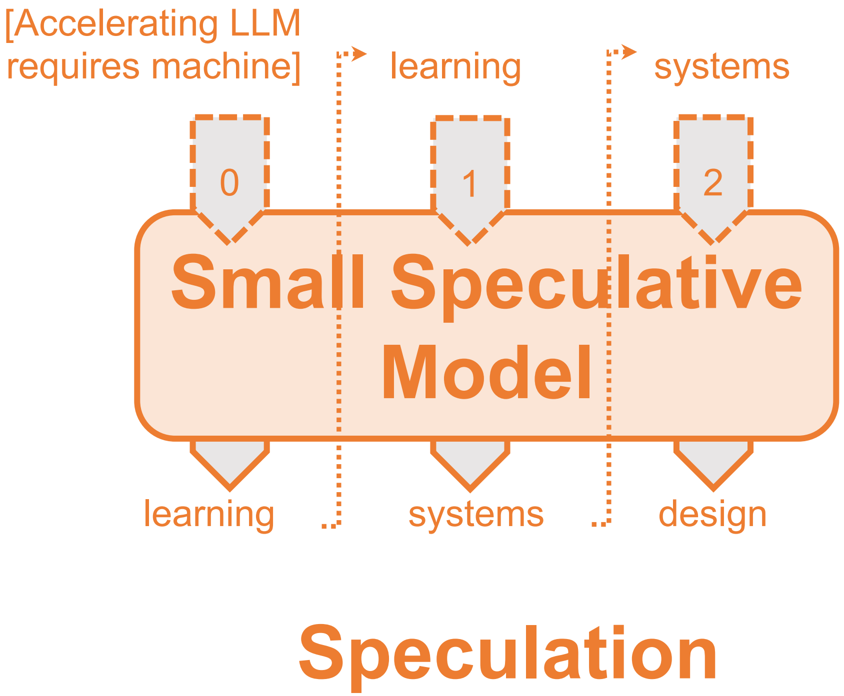
Pro: cheap and fast

Con: less accurate

* Language Models are Few-Shot Learners. Arxiv. 2005.14165

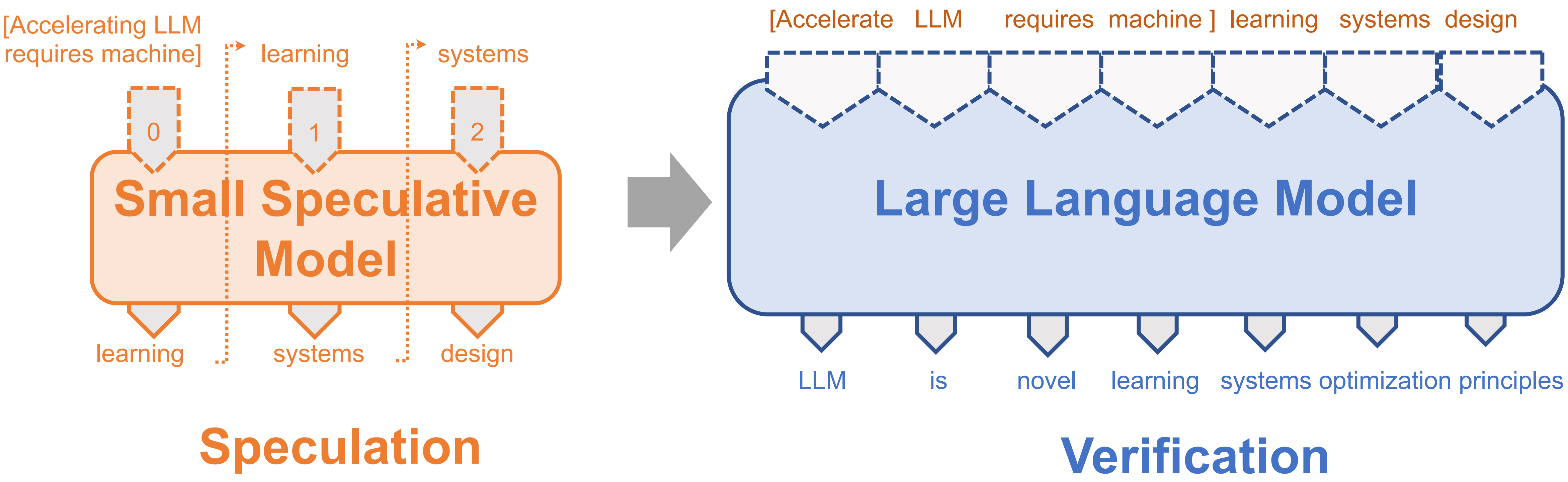
Speculative Decoding

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

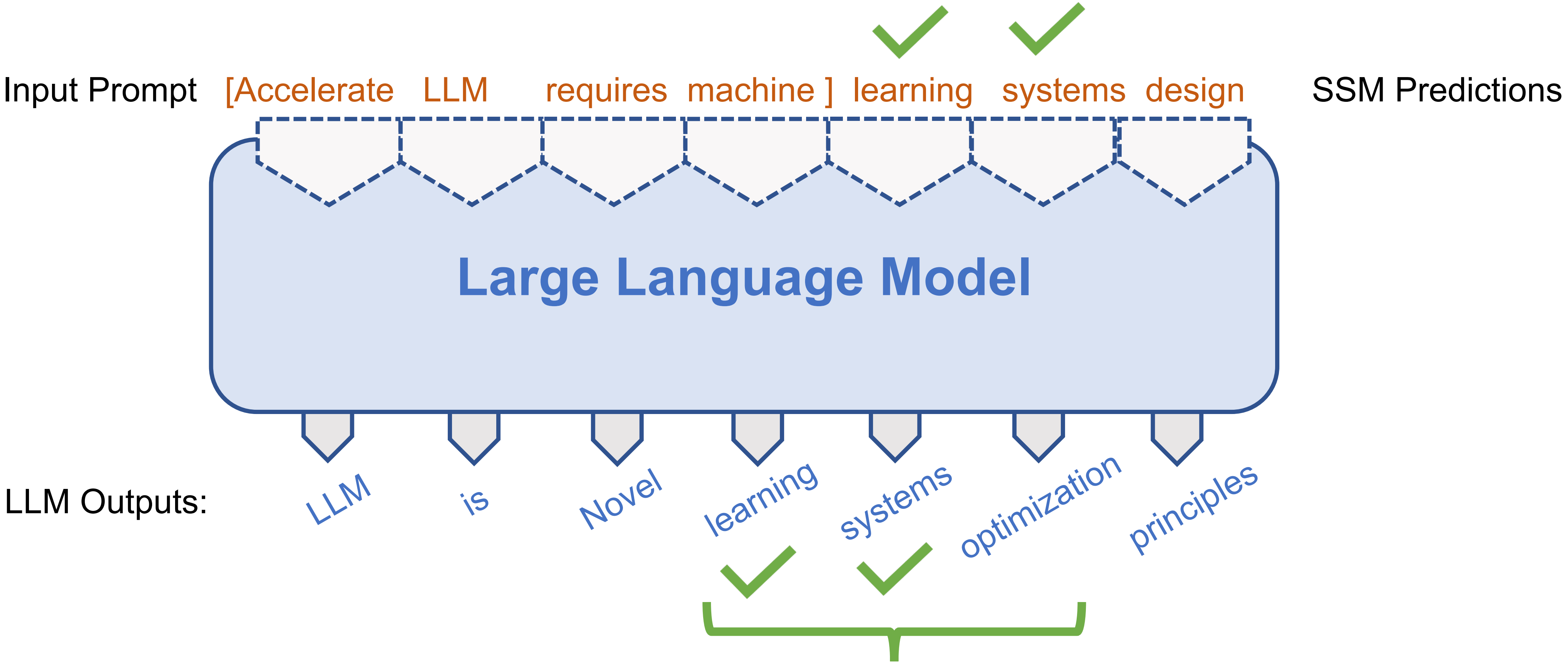


Speculative Decoding

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

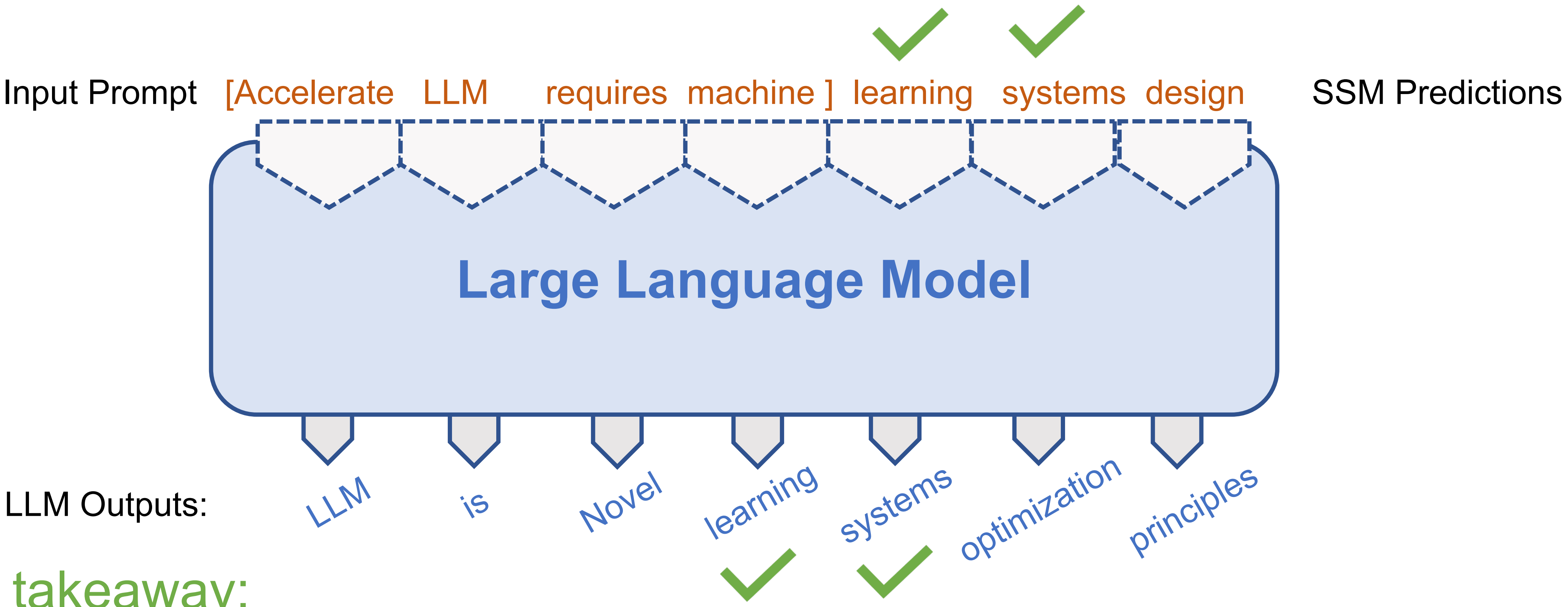


Verifying Speculative Decoding Results



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

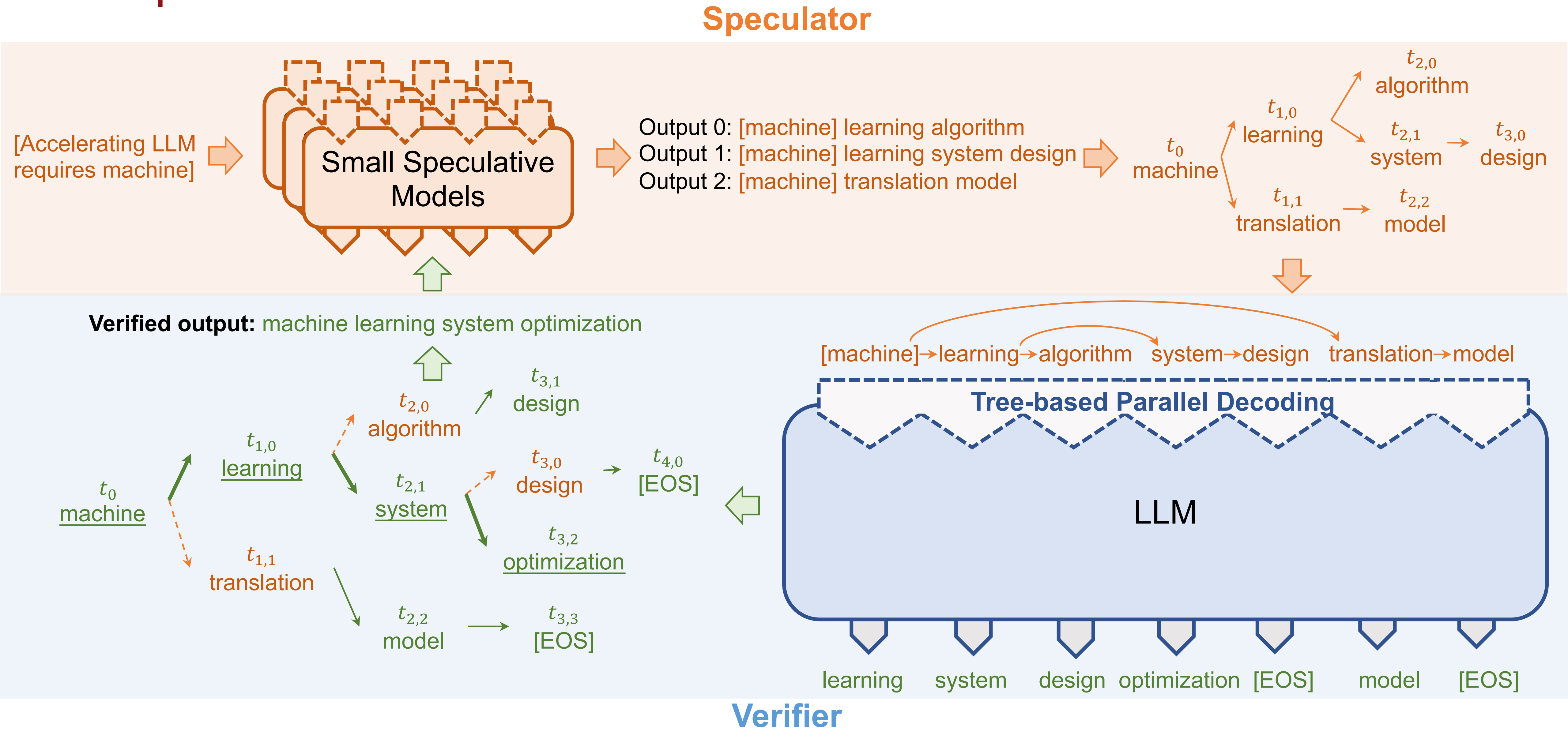
SpecInfer: Tree-based Speculative Inference & Verification

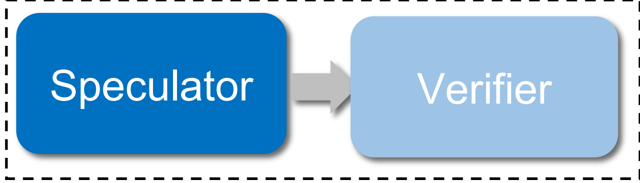
Key idea: not use LLMs as incremental decoder, use them as **parallel token tree verifier**

- **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

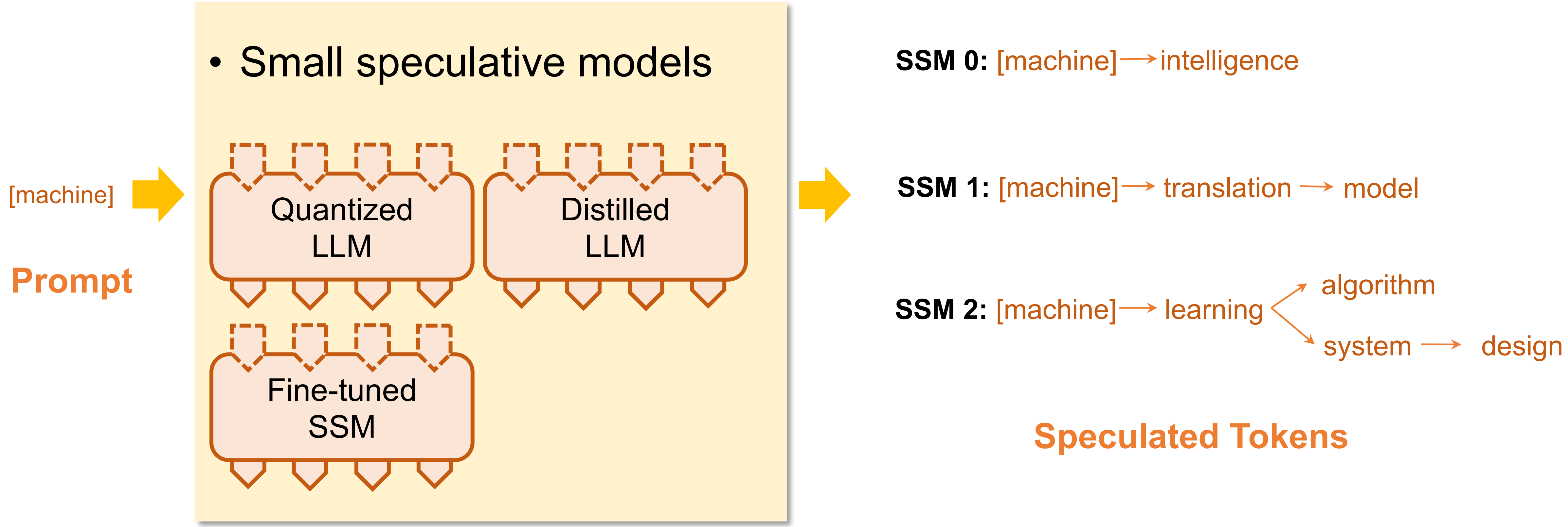
* SpecInfer: Accelerating LLM Serving with Tree-based Speculative Inference and Verification. ASPLOS'24

SpecInfer Workflow

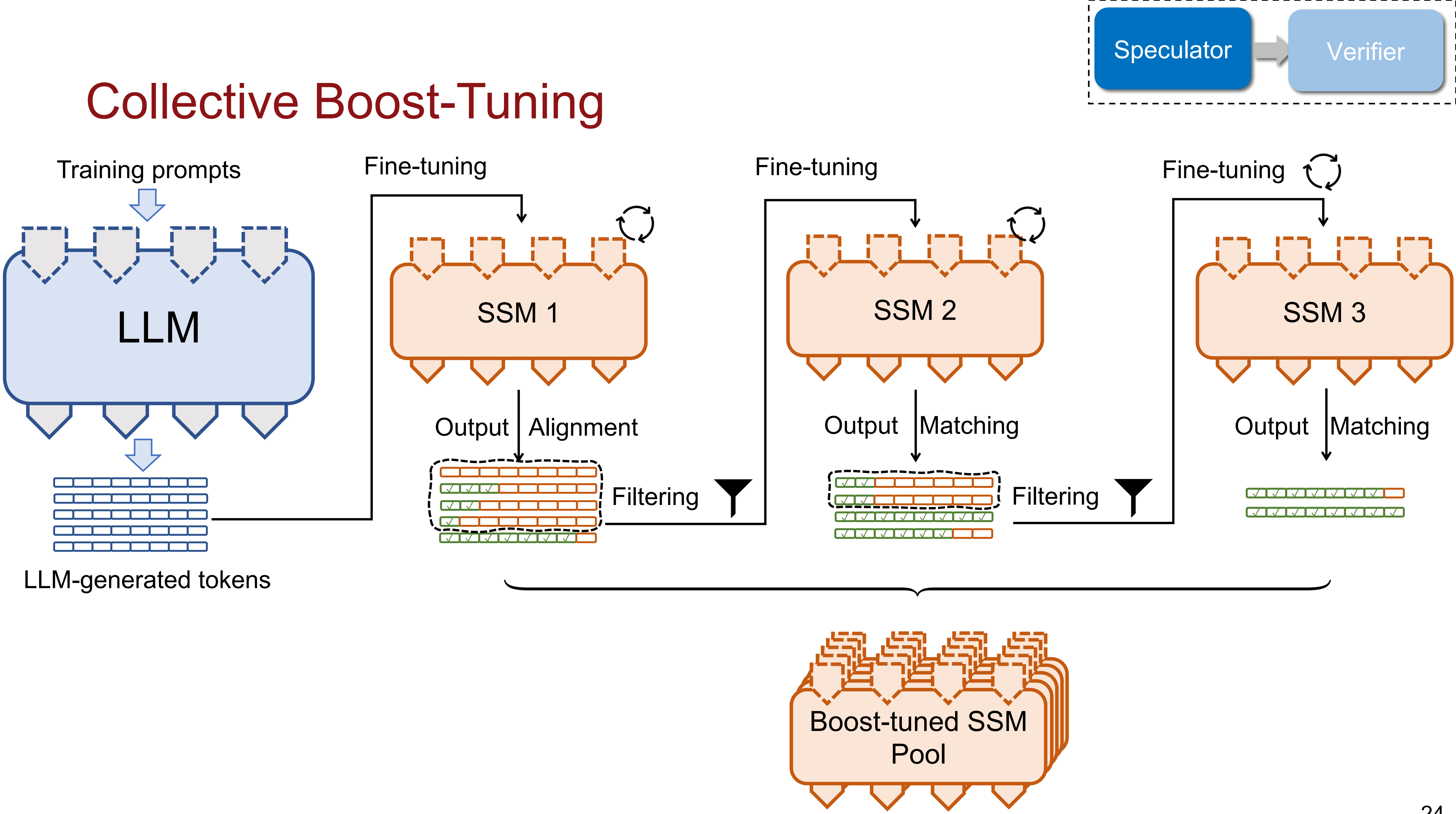




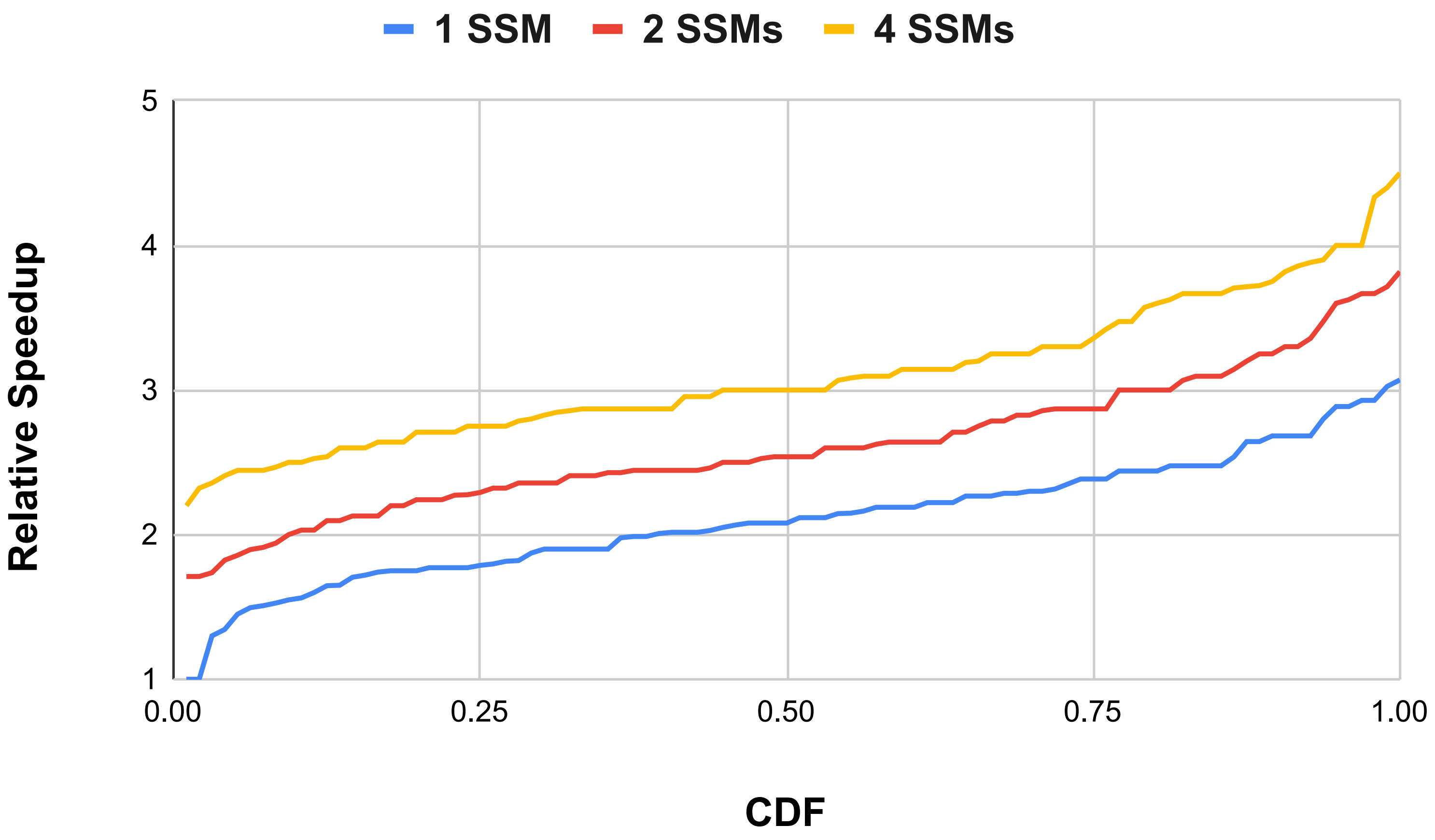
Learning-based Speculator



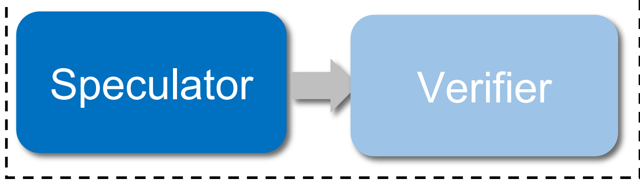
Collective Boost-Tuning



Collective Boost-Tuning Consistently Improves Performance



* LLM: LLAMA-7B, SSM: LLAMA-160M, dataset: Alpaca, GPU: NVIDIA A10

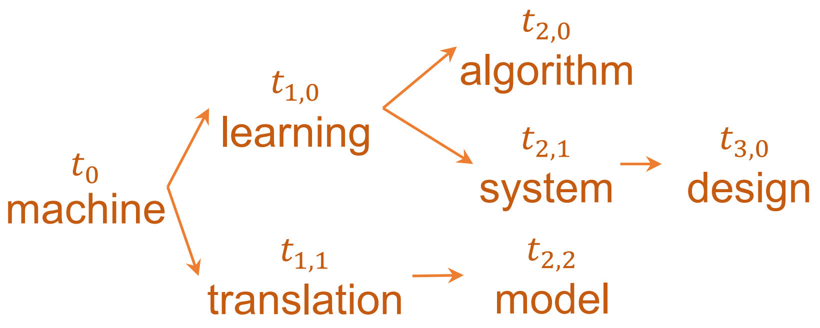


Token Tree Merge

- A compact way to represent speculated tokens

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

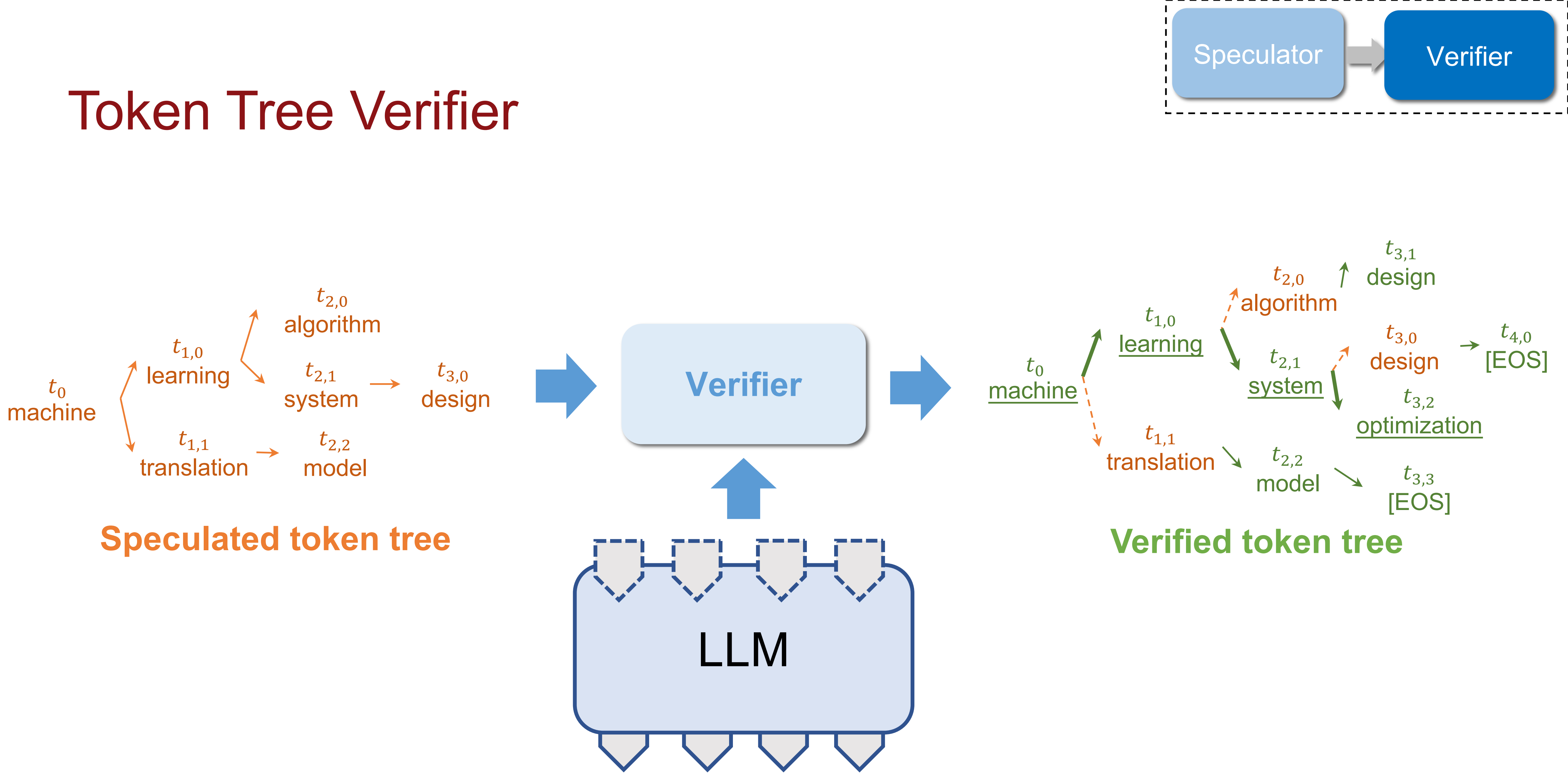
Tree Merge

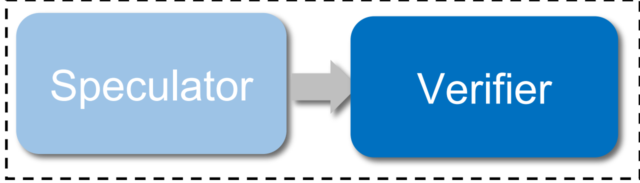


Token Sequences

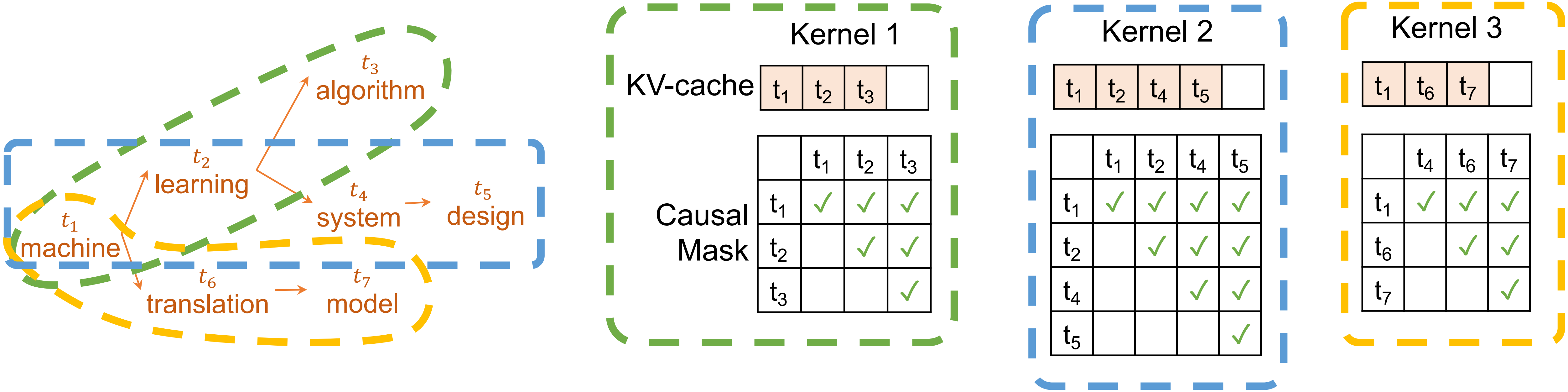
Token Tree

Token Tree Verifier



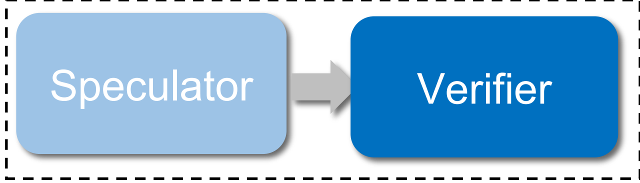


Sequence-based Decoding



Issues:

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



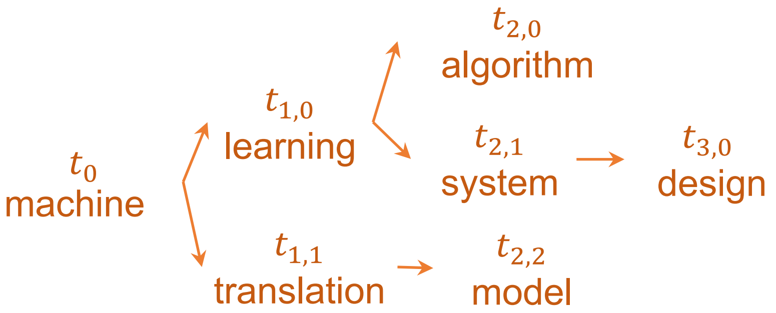
Tree Attention

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

Token Sequences

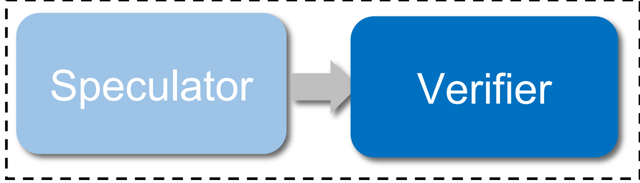
Attention([machine] learning algorithm
[machine] learning system design
[machine] translation model)

= Tree-Attention(



)

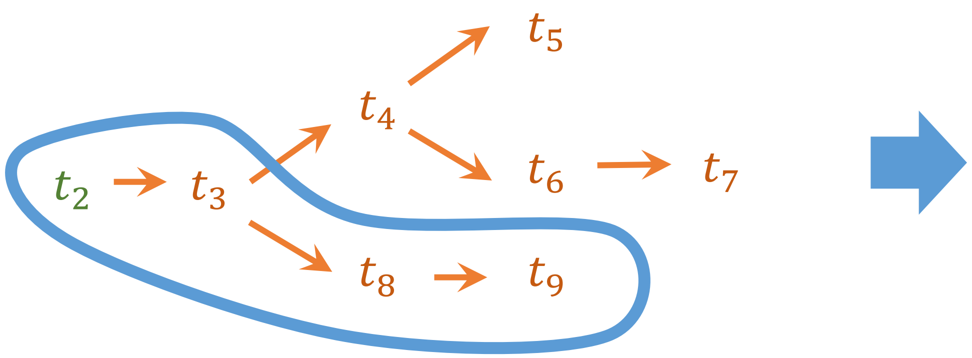
Token Tree



Tree-based Parallel Decoding

KV-cache

t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	…



Topology-aware causal mask

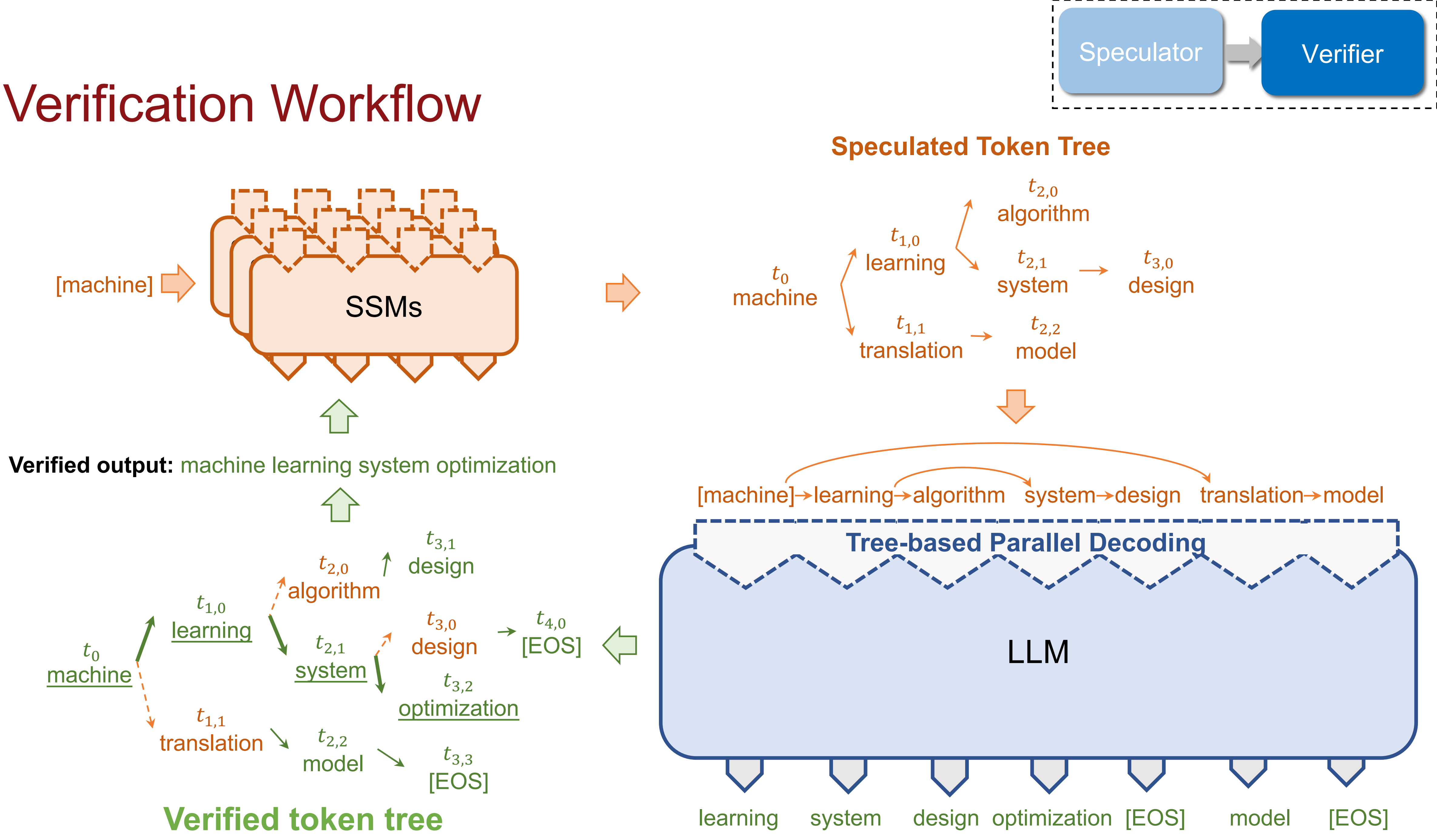
	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
t_2	✓	✓	✓	✓	✓	✓	✓	✓
t_3		✓	✓	✓	✓	✓	✓	✓
t_4			✓	✓	✓	✓		
t_5				✓				
t_6					✓	✓		
t_7						✓		
t_8							✓	✓
t_9								✓

Key optimizations:

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

* SpecInfer: Accelerating LLM Serving with Tree-based Speculative Inference and Verification. ASPLOS'24

Verification Workflow



Stochastic Decoding

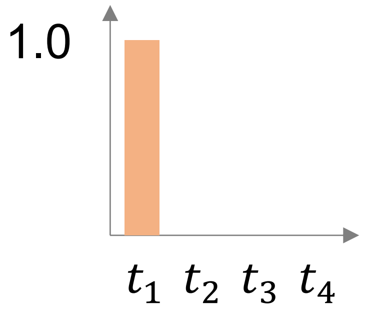
- **Challenge:** verifying stochastic equivalence

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

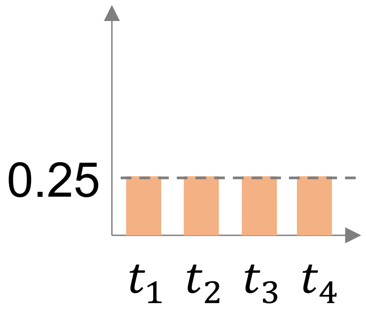
- A strawman approach: **naïve sampling**
  - Use LLM to sample $x_i \sim P_{\text{IncrDecode}}(\cdot\,|x_{<i}, \text{LLM})$
  - Verify if x_i is in the token tree

Naïve Sampling can be Suboptimal

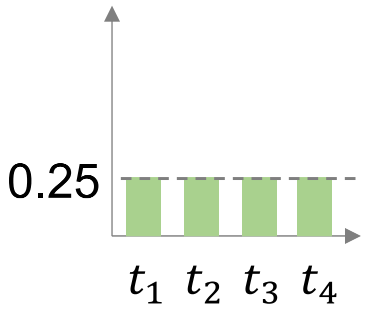
- Assume one LLM, two SSMs, and four possible tokens: t_1, t_2, t_3, t_4



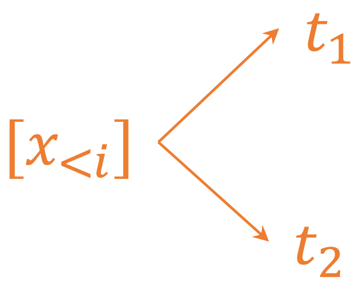
SSM 1: $P(\cdot \mid x_{<i}, \mathrm{SSM}_1)$



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



LLM: $P(\cdot \mid x_{<i}, \mathrm{LLM})$



Token Tree

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

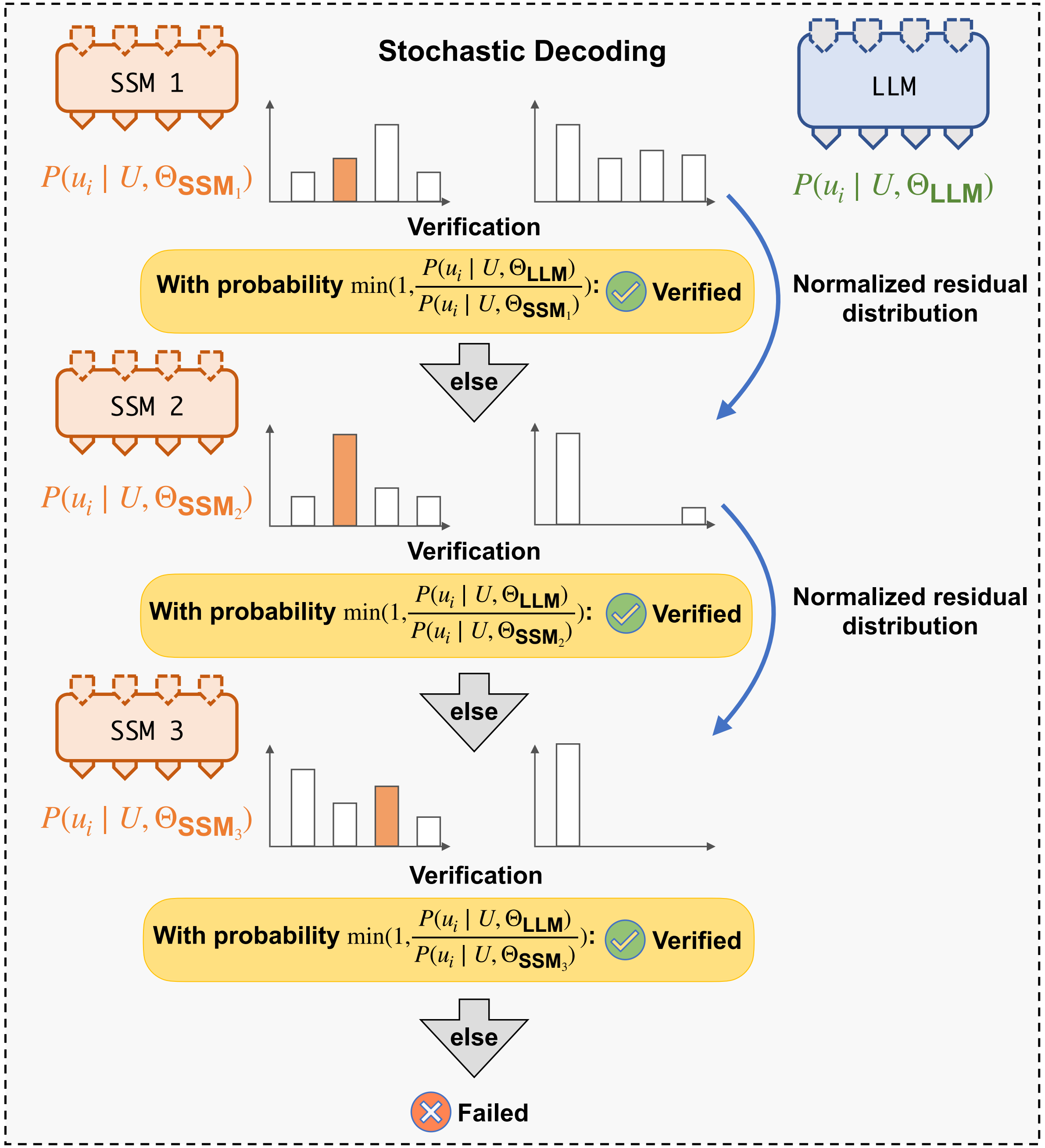
But we can do better by directly accepting SSM 2; verification prob. = 100%

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

Speculative Sampling

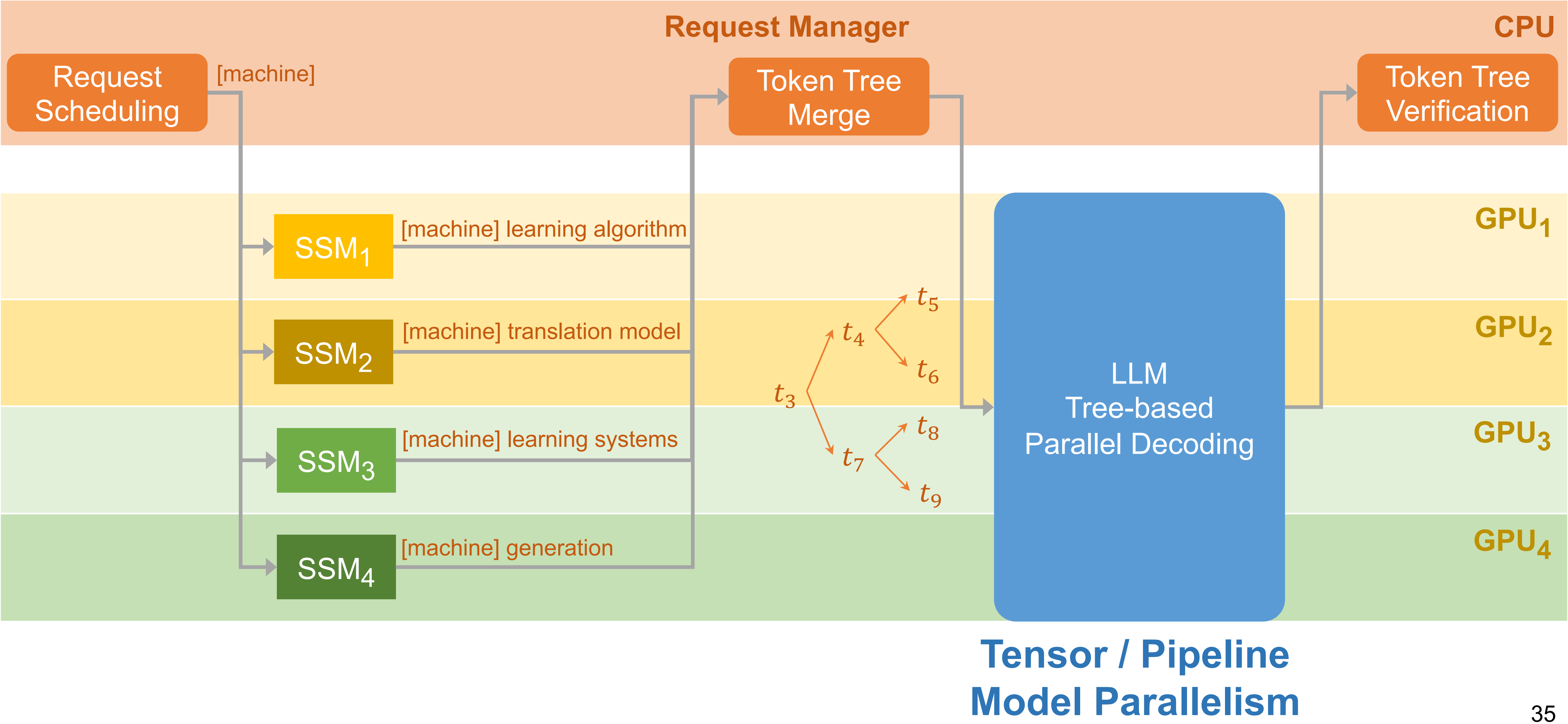
1. Sample a token $x \sim P(u_i|U, \Theta_{SSM})$ using SSM
2. If $P(x|U, \Theta_{SSM}) \leq P(x|U, \Theta_{LLM})$, directly accept x
3. If $P(x|U, \Theta_{SSM}) > P(x|U, \Theta_{LLM})$, accept x with prob. $\frac{P(x|U,\Theta_{LLM})}{P(x|U,\Theta_{SSM})}$
4. If reject x, normalize residual distribution

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

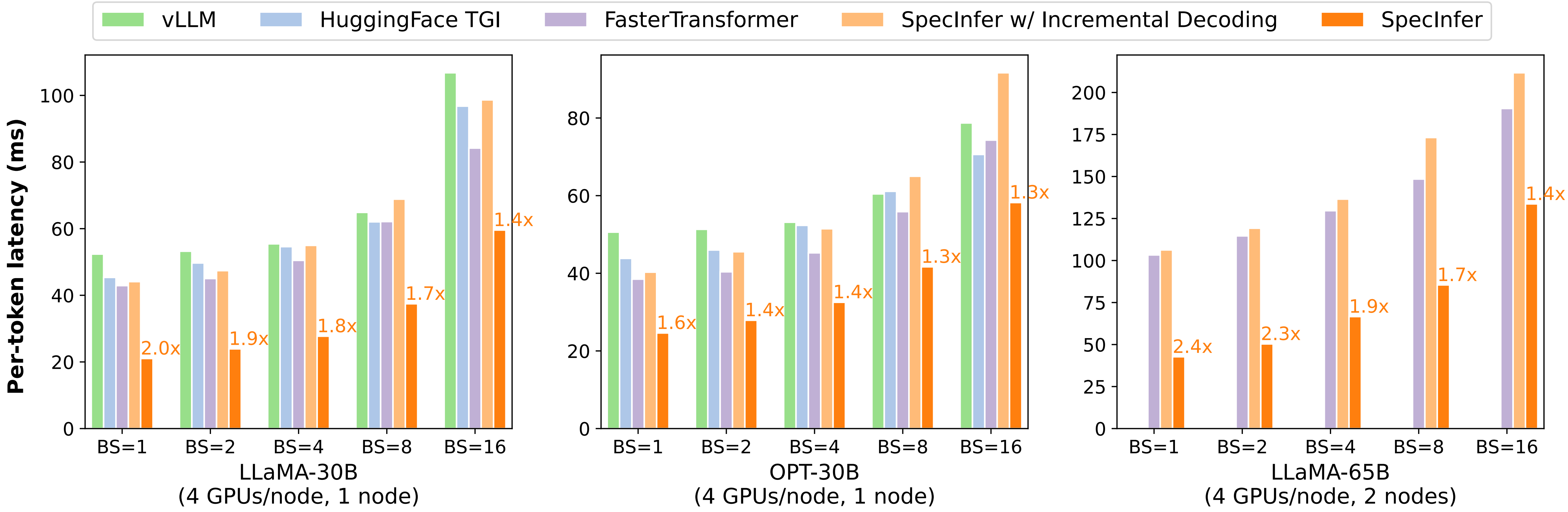


* Details in SpecInfer: Accelerating Large Language Model Serving with Tree-based Speculative Inference and Verification. ASPLOS'24

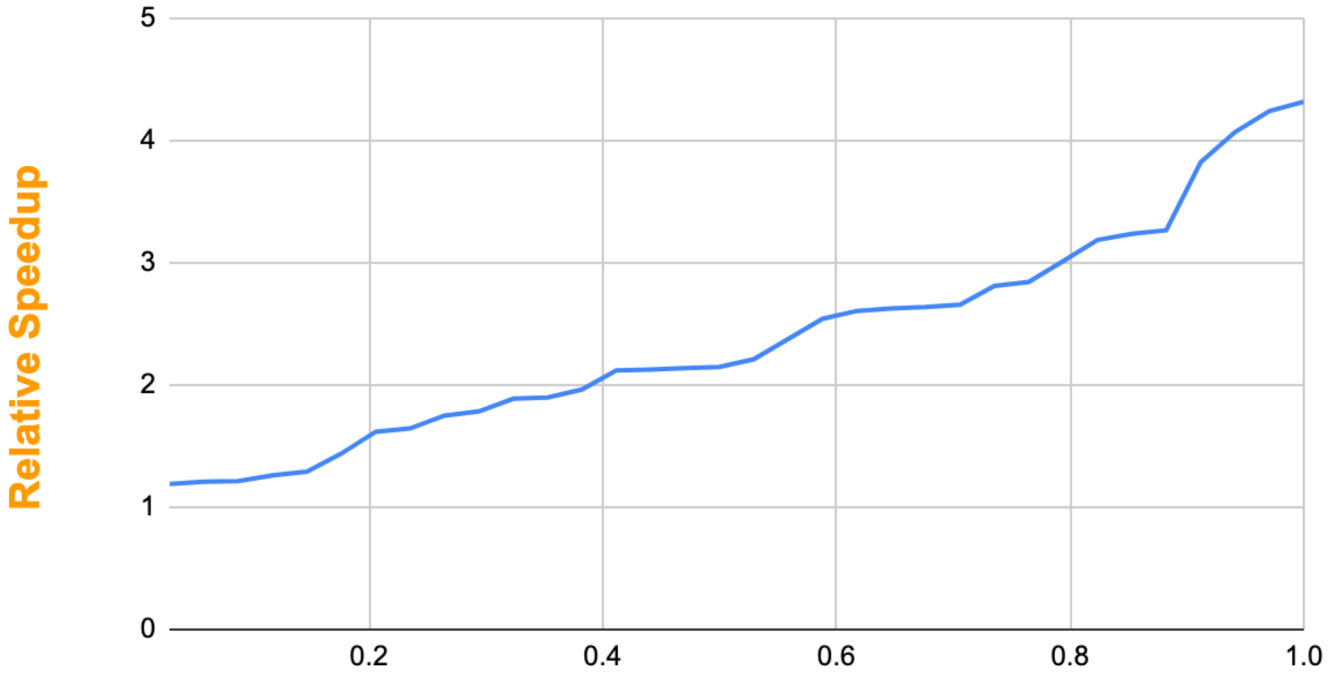
Distributed LLM Serving



SpecInfer Accelerates LLM Inference by 1.3-2.4x

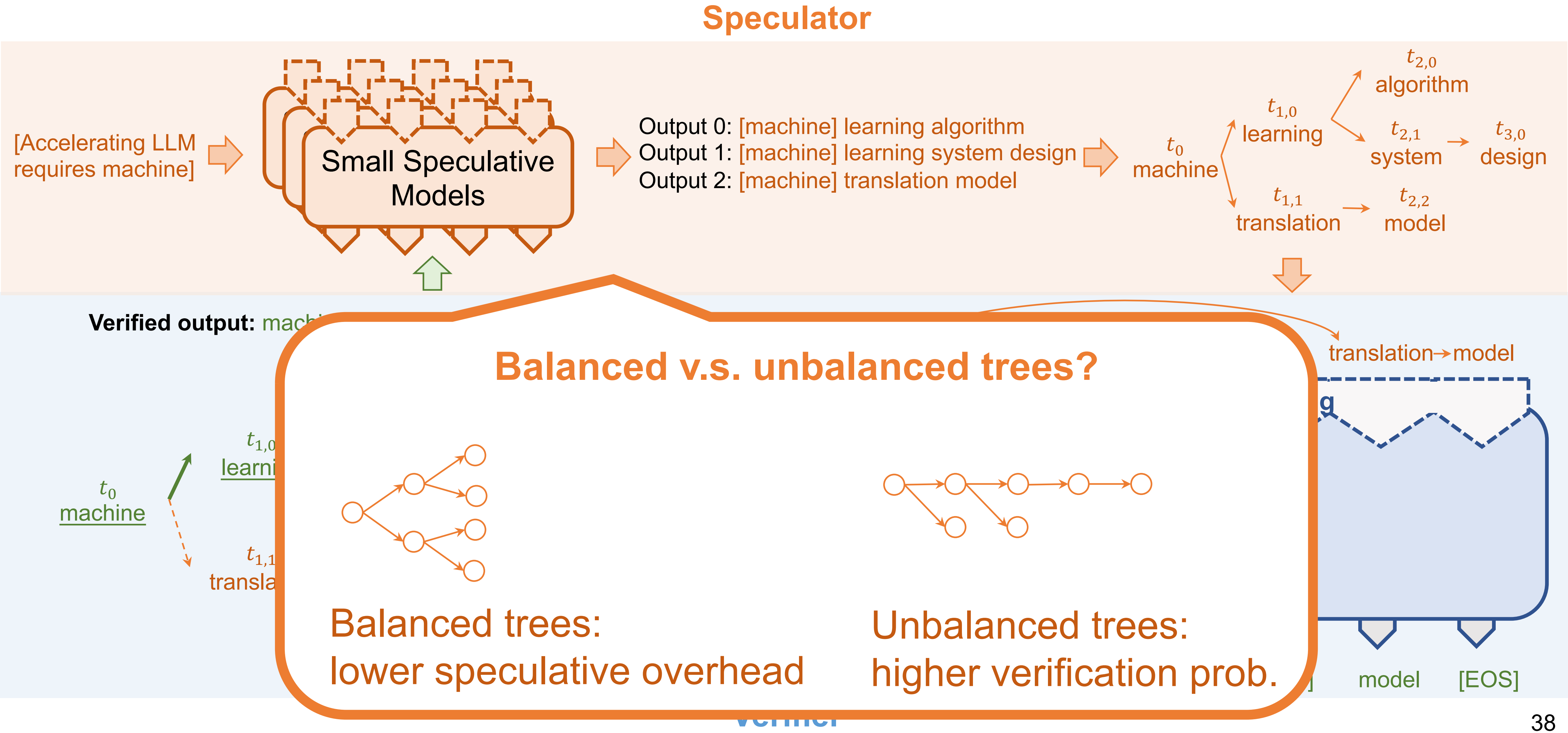


SpecInfer can Consistently Accelerate LLM Inference

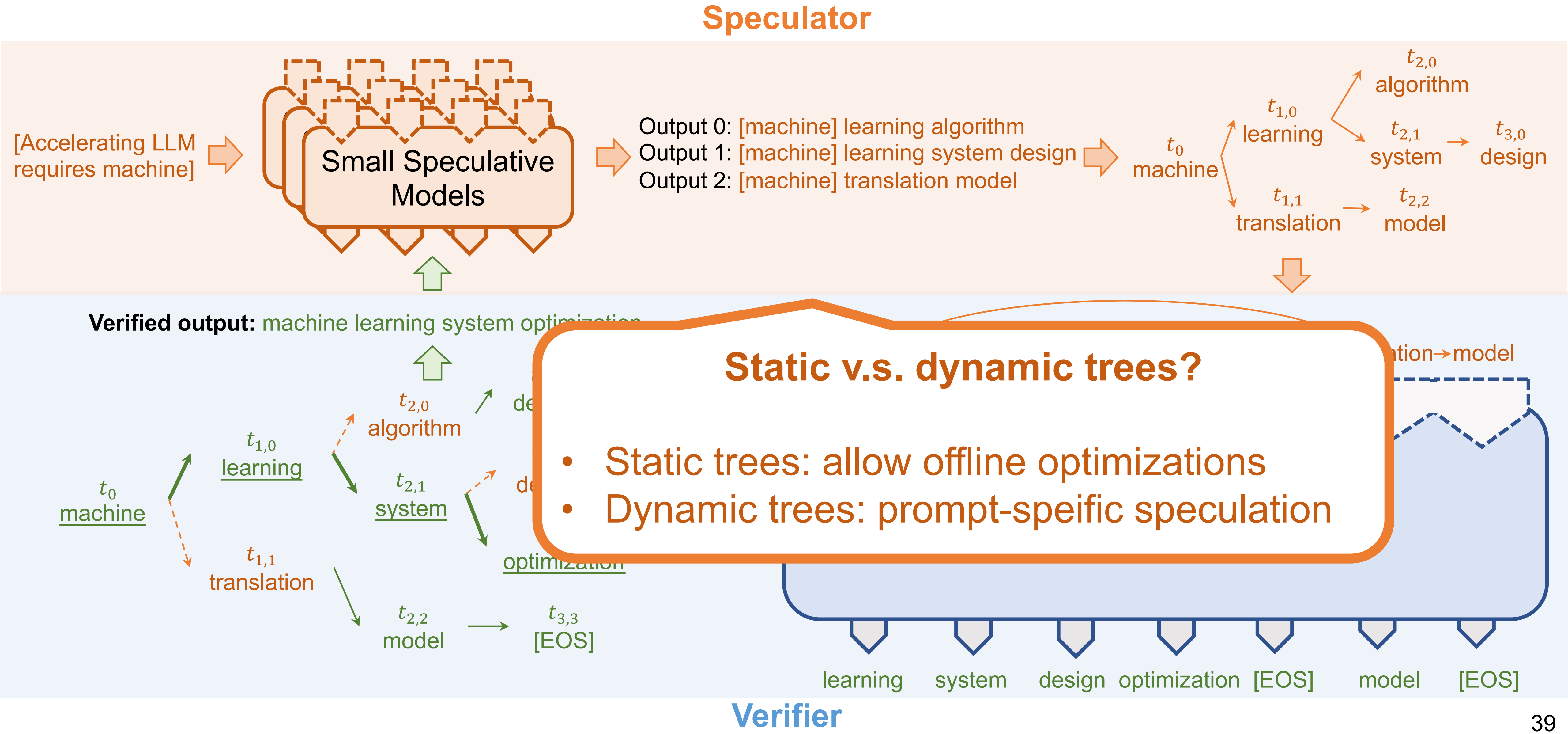


LLM: LLaMA-65B, SSMs: LLaMA-160M

Open Research Questions



Open Research Questions



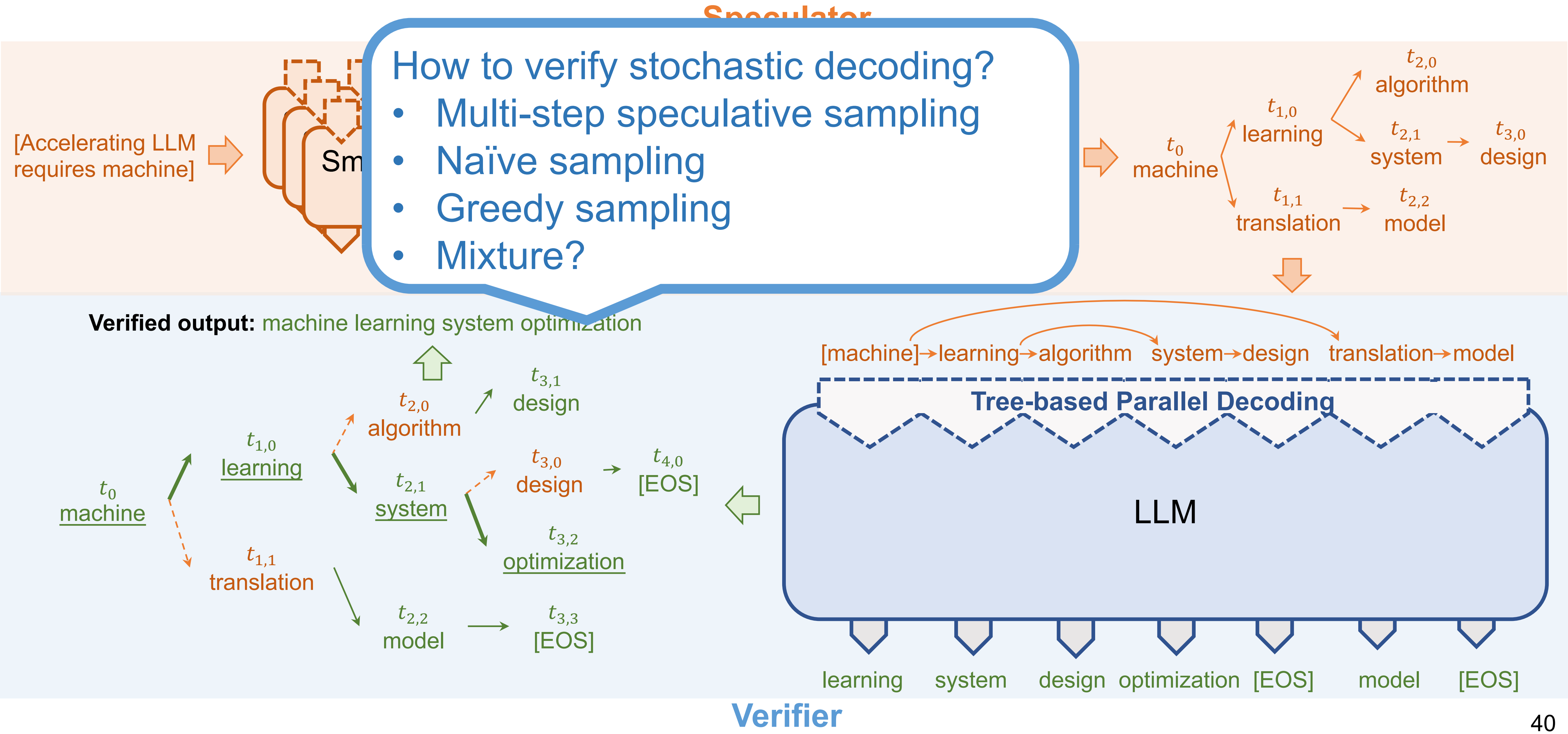
Static v.s. dynamic trees?

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

Open Research Questions

How to verify stochastic decoding?

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



Recap: LLMs Serving Techniques

- Continuous Batching
- Speculative Decoding