FY2025/2025年度

AI Computing Team, Shinya Takamaeda

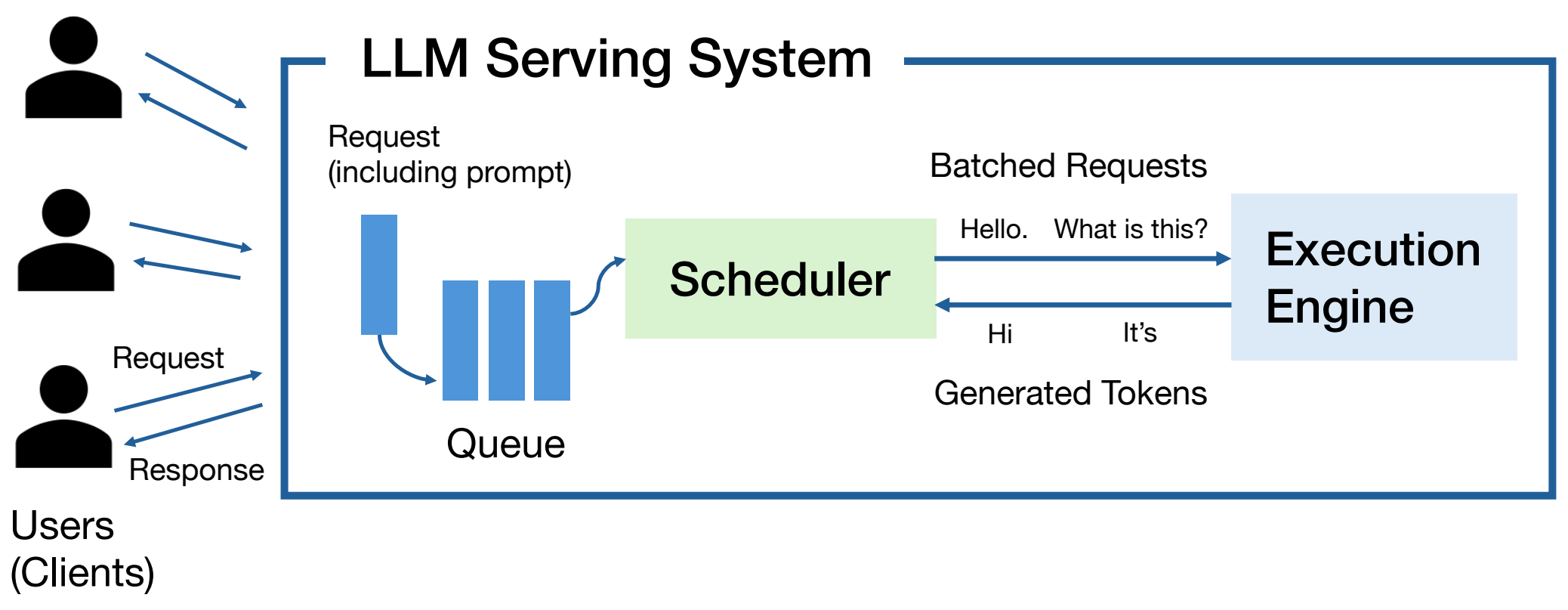
AIコンピューティングチーム 高前田 伸也

RIKEN

Center for Advanced Intelligence Project

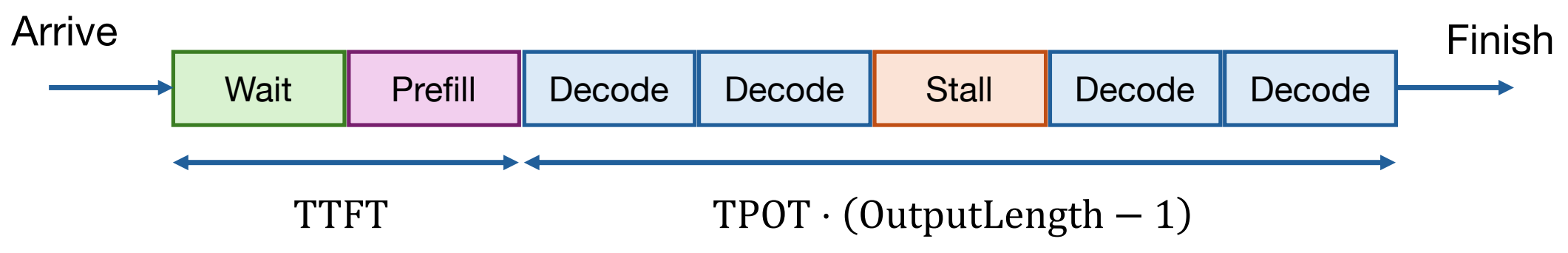
Adaptive Resource Allocation for Low-Latency LLM Serving

LLM Serving Systems (ChatGPT, etc.)



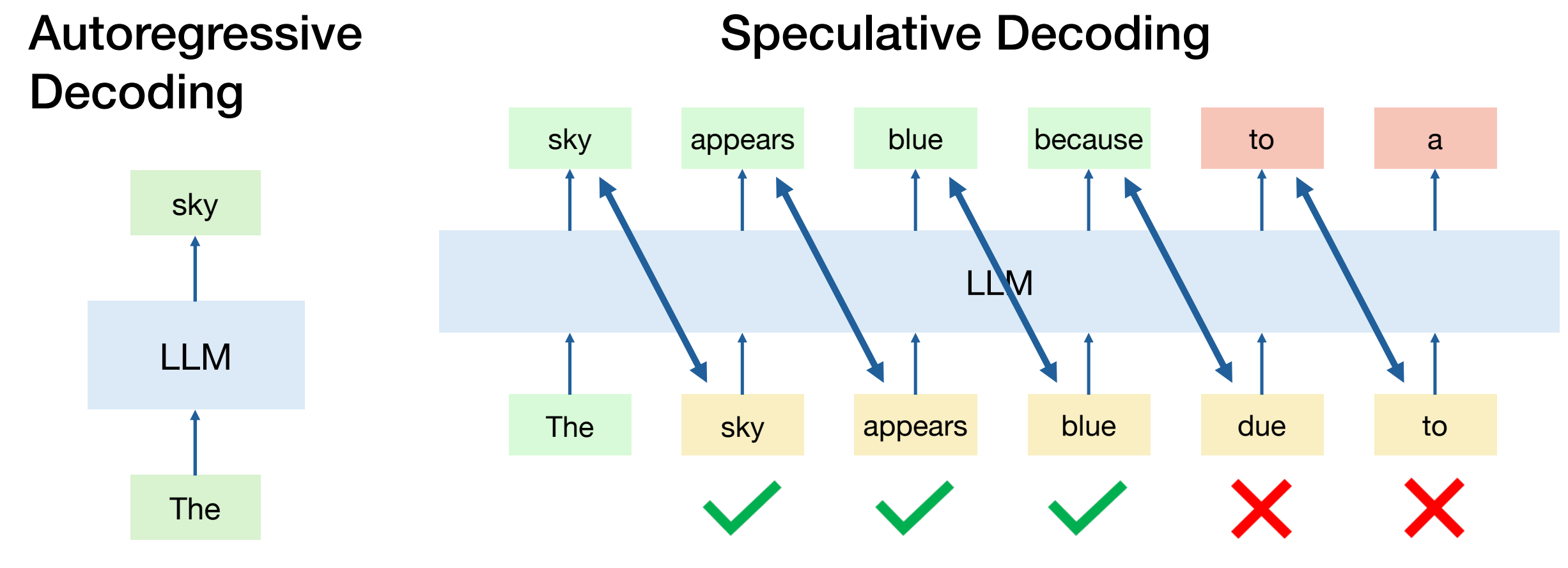
Performance Metrics

- Time to First Token (TTFT) ↓
- Time per Output Token (TPOT) ↓



Speculative Decoding

A small **draft model** generates candidate output tokens speculatively and the **target model** verifies the candidates for reducing **TPOT**

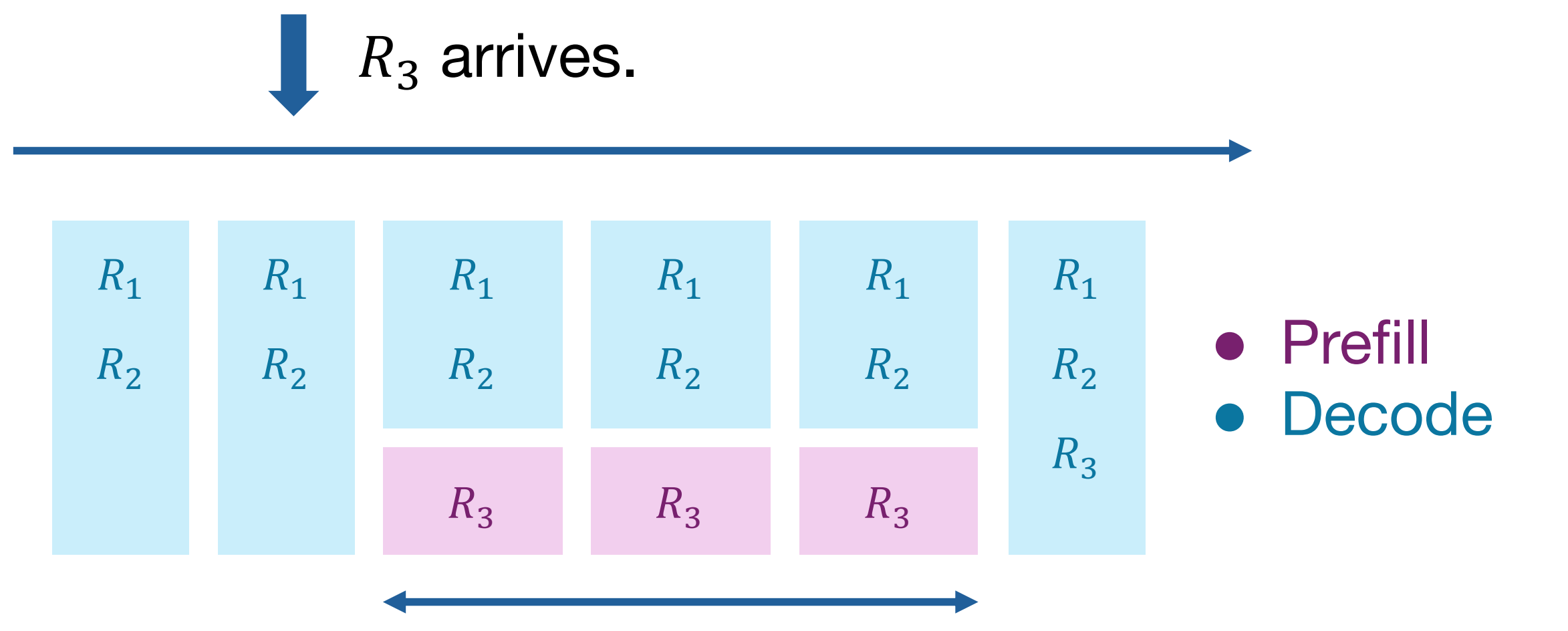


A model only processes one token per iteration

A model processes multiple tokens per iteration, increasing arithmetic intensity for better perf

Chunked Prefill

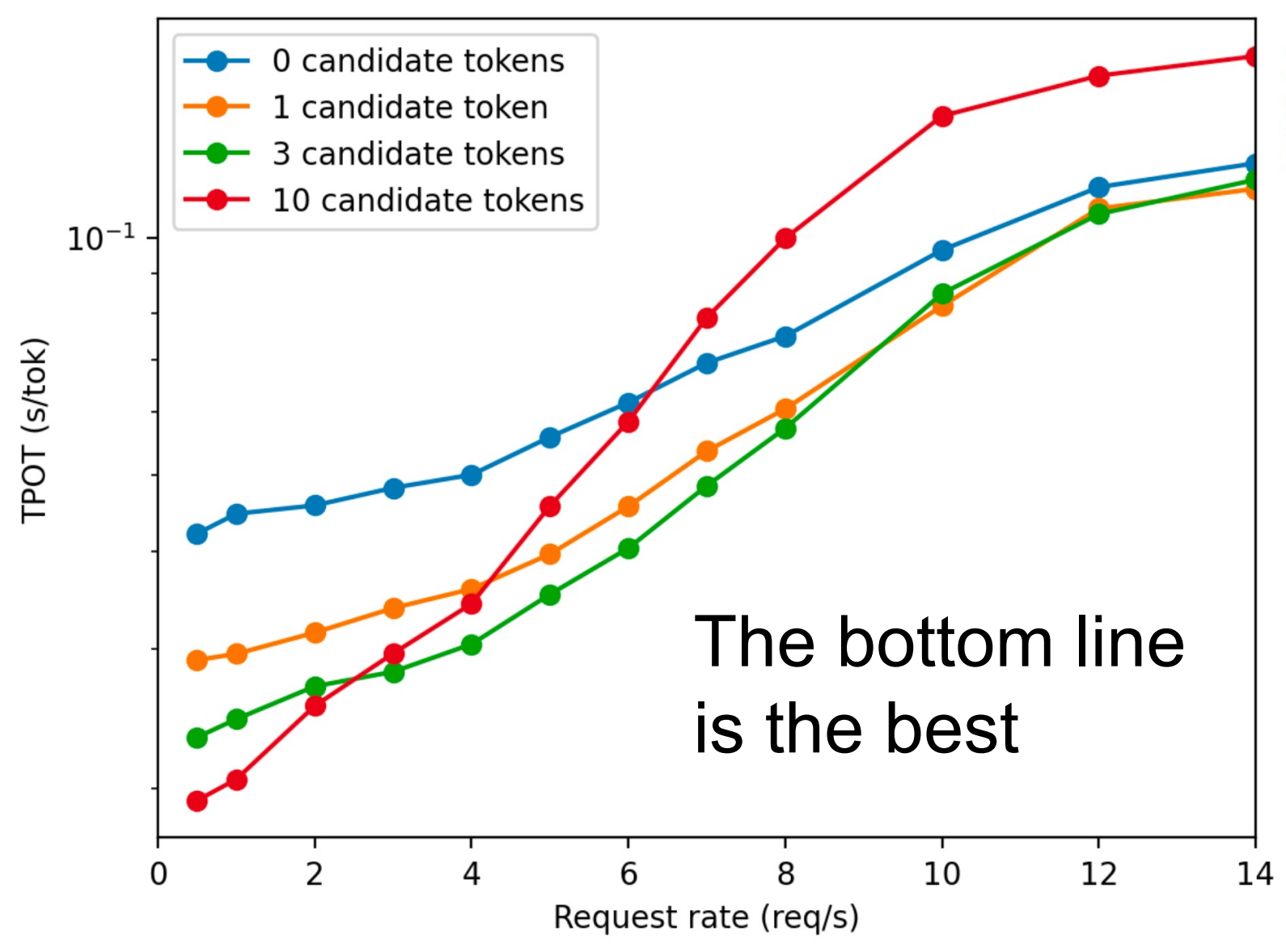
Mixing the prefill processing of a new request with earlier decode requests for reducing TTFT of the new request



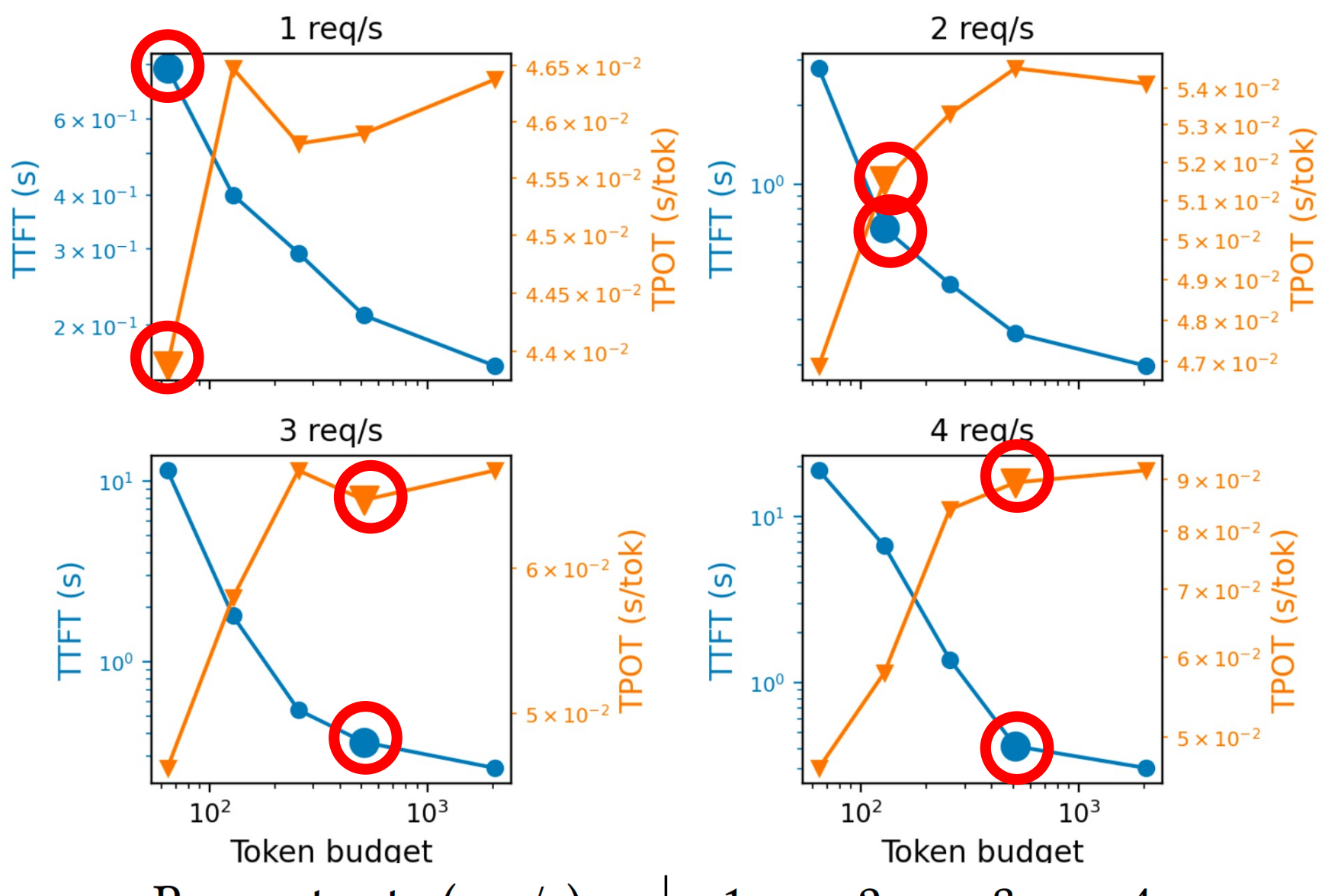
Optimizing the number of candidate tokens in Speculative Decoding and the token budget in Chunked Prefill

Optimal aggressiveness of Speculative Decoding and Chunked Prefill depends on the request rate (request/sec).

We propose an adaptive resource allocation method tailored to these acceleration techniques for low TPOT.



Speculative Decoding



Request rate (req/s)	1	2	3	4
Optimal token budget	64	128	512	512

Chunked Prefill

Minimizing TPOT:

$$\underset{c \geq 0}{\text{minimize}}\ \text{TPOT}$$

where *c* donates the number of candidate tokens per request.

TPOT is defined as follows:

$$\begin{aligned}\text{TPOT} &= \frac{\text{DecodeTime}}{\text{OutputLength}-1} \\ &= \frac{\text{DecodeTime}/\text{NumIters}}{(\text{OutputLength}-1)/\text{NumIters}} \\ &= \frac{\text{PerIterExecTime}}{\text{PerIterGenTokens}}\end{aligned}$$

We obtain the following optimization problem:

$$\underset{c \geq 0}{\text{maximize}}\ \frac{\text{PerIterGenTokens}}{\text{PerIterExecTime}}$$

Minimizing TPOT under the token budget constraint:

$$\underset{n \geq 0}{\text{minimize}}\ \text{TPOT} \quad \text{subject to TTFT} \leq l$$

where *n* donates the token budget and *l* donates the given target TTFT value (i.e., latency requirement).

We formulate the constraint as follows:

$$n < \sum_{j=1}^{i} n_j \implies \frac{n}{\text{PerIterExecTime}} \geq \frac{\sum_{j=1}^{i} n_j}{(t_i + l) - t} \quad \text{for all } i = 1, \ldots, m.$$

where n_j donates the number of remaining prefill tokens for request R_j and t_i donates the arrival time of request R_j.

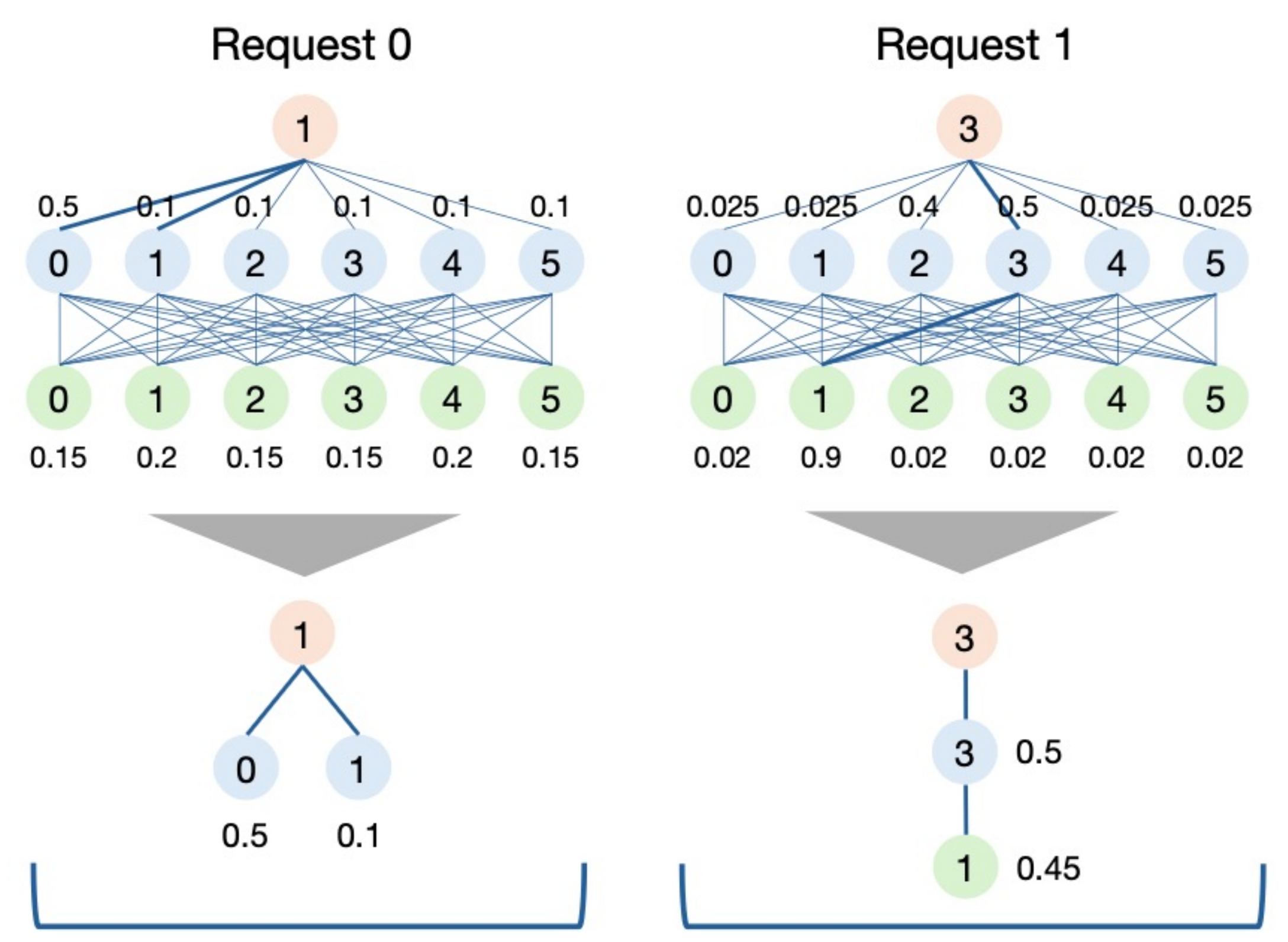
Implementation

- **PerIterExecTime** is predicted by a lookup table based on **Extremely Randomized Trees.**
- **PerIterGenTokens** is predicted by a lookup table based on prior measurements of the number of generated tokens **under the different numbers of candidate tokens.**

Batch-Aware Speculative Decoding

We also propose the global optimization of draft tree structure, which allocates speculation resources for each request according to the acceptance probability.

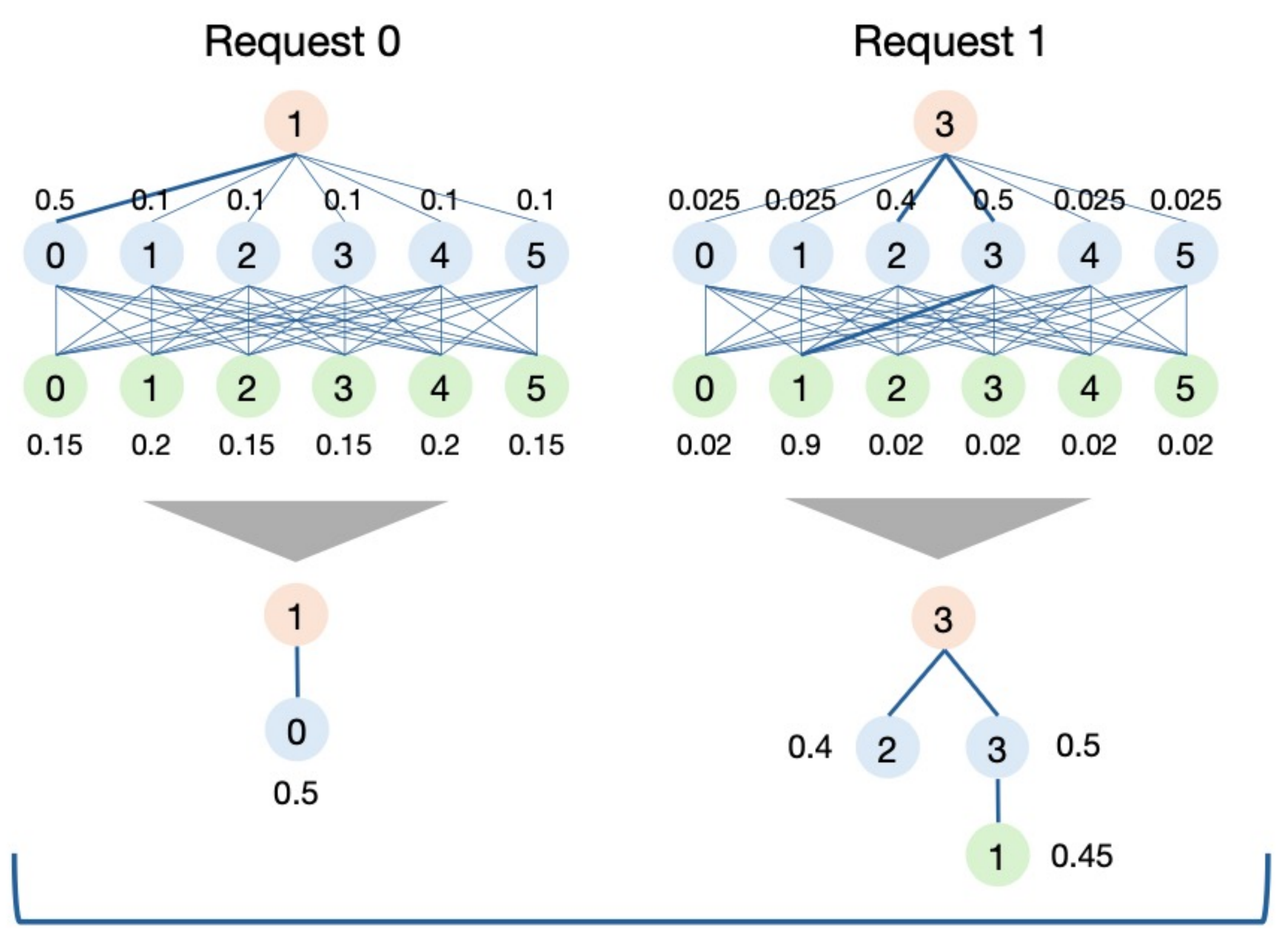
(A) Local optimization of draft tree structures



2 candidate tokens

2 candidate tokens

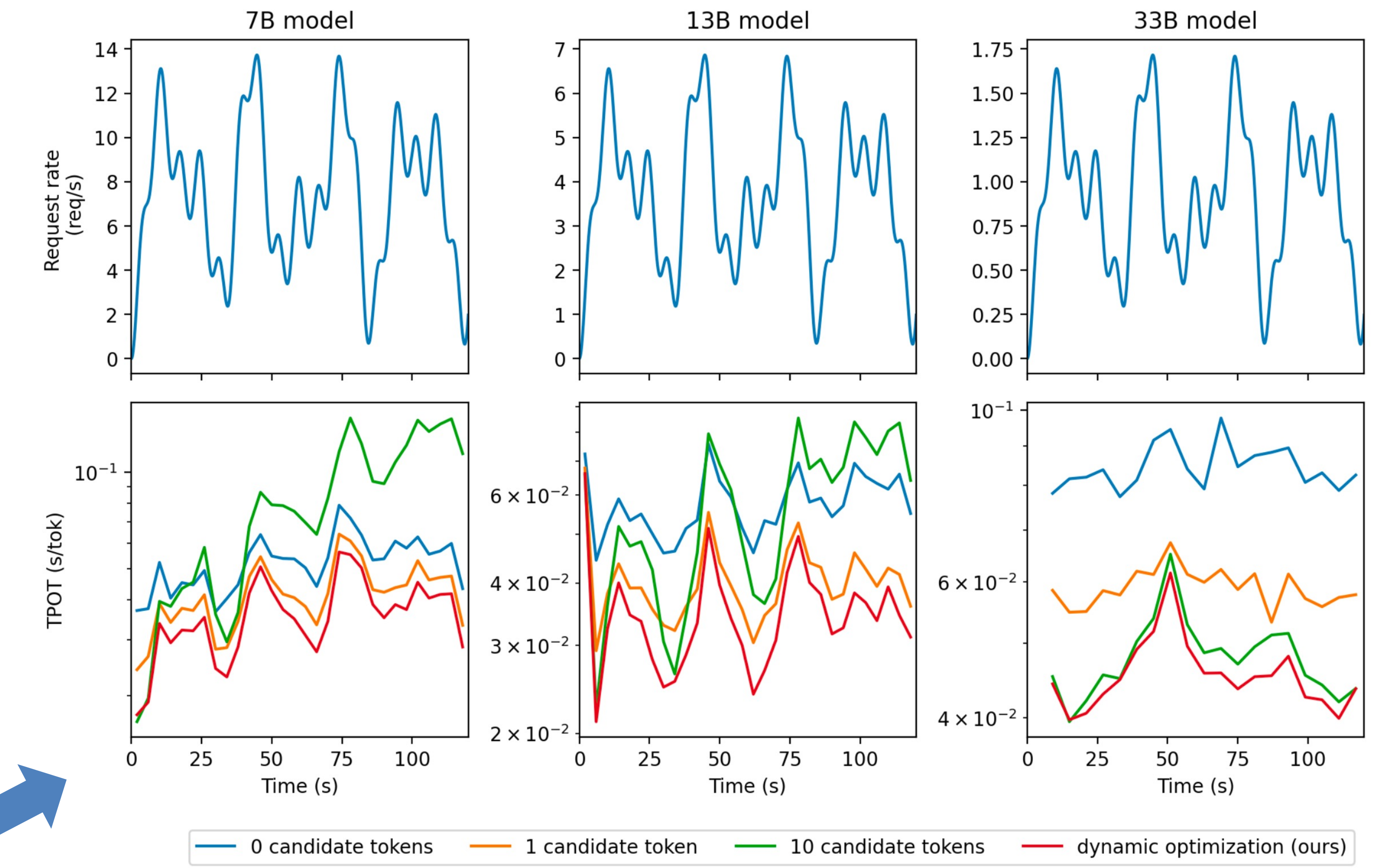
(B) Global optimization of draft tree structures



4 candidate tokens (in total)

Experimental Setup

- Model: Vicuna 7B, 13, and 33B
- GPU: NVIDIA A100 80GB
- Workload: LMSYS-Chat-1M, CNN/DailyMail, BigCodeBench
- Requests are generated based on synthetic request probability

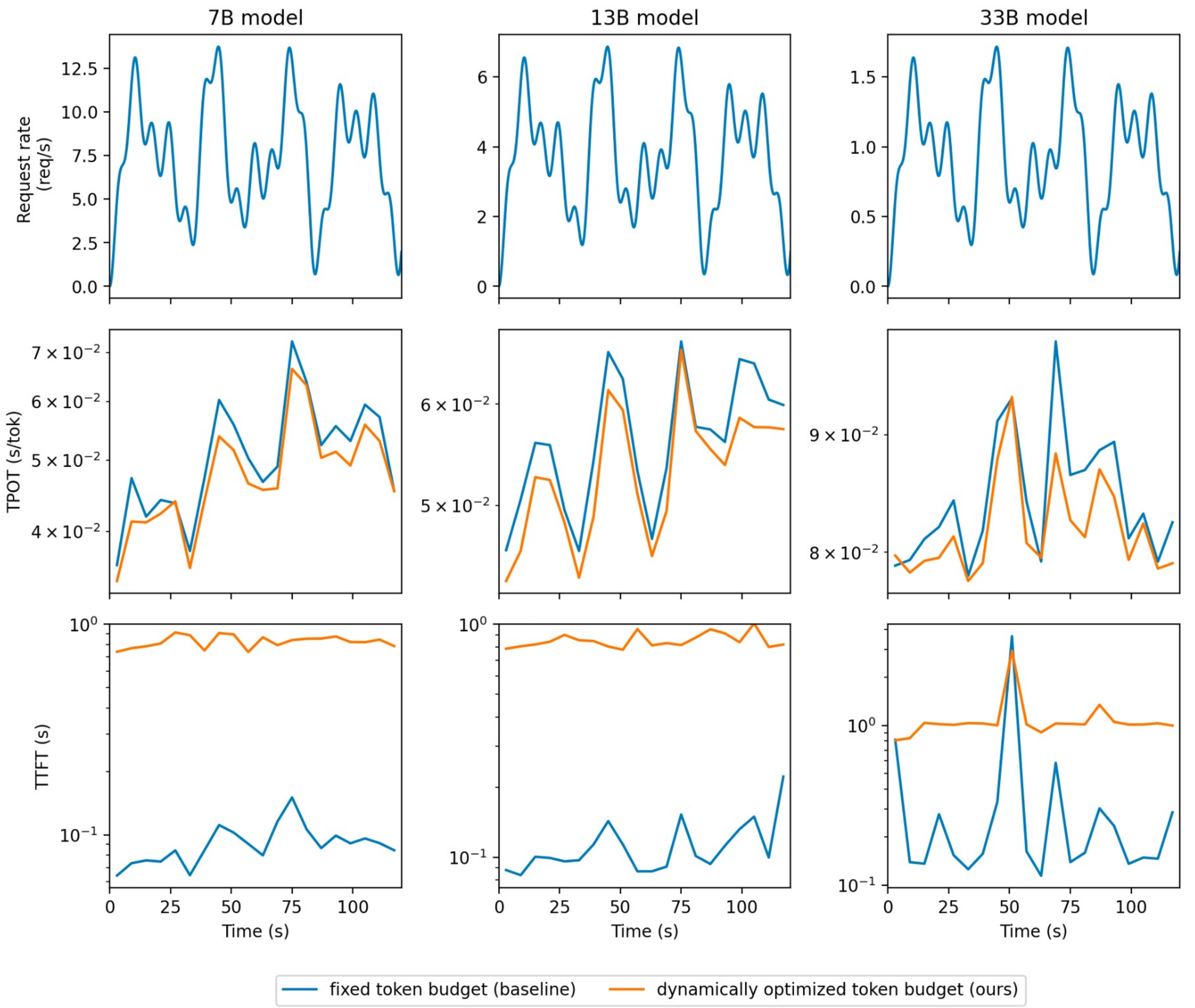


For Speculative Decoding

- A single fixed number of candidate tokens cannot achieve the lowest TPOT (latency) not for all time
- Our dynamic optimization achieves lowest TPOT (average latency) for all time

For Chunked Prefill

- Our dynamic optimization achieves lower TPOT than the fixed token budget, while both satisfies the TTFT (latency) requirement.


7B model
13B model
33B model
Request rate (req/s)
TPOT (s/tok)
TTFT (s)
Time (s)
fixed token budget (baseline)
dynamically optimized token budget (ours)