

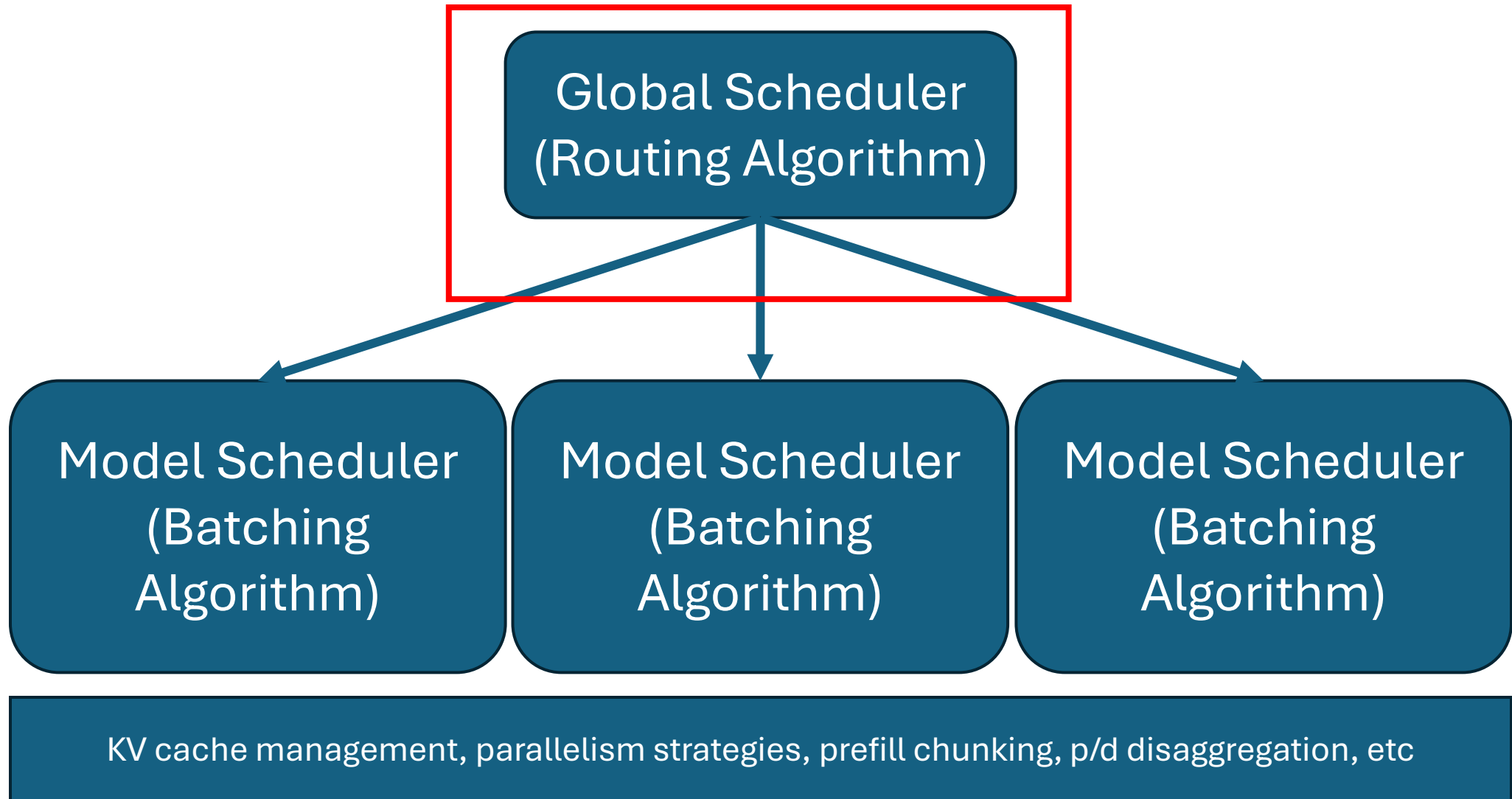
Performance Aware LLM Load Balancer for Mixed Workloads

Kunal Jain, Anjaly Parayil, Ankur Mallick, **Esha Choukse**, Xiaoting Qin, Jue Zhang, Íñigo Goiri, Rujia Wang, Chetan Bansal, Victor Rühle, Anoop Kulkarni, Steve Kofsky, Saravan Rajmohan

The 5th Workshop on Machine Learning and Systems (EuroMLSys)
March 31, 2025



LLM Serving frameworks

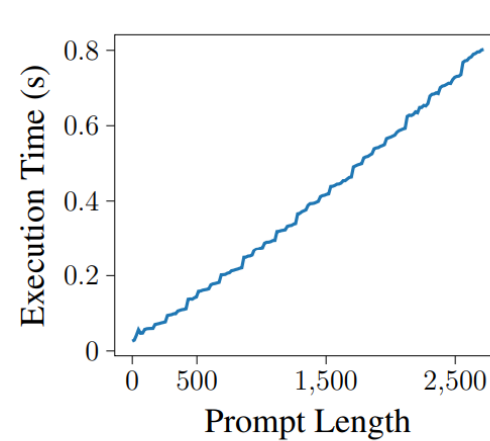


What does a router need to decide?

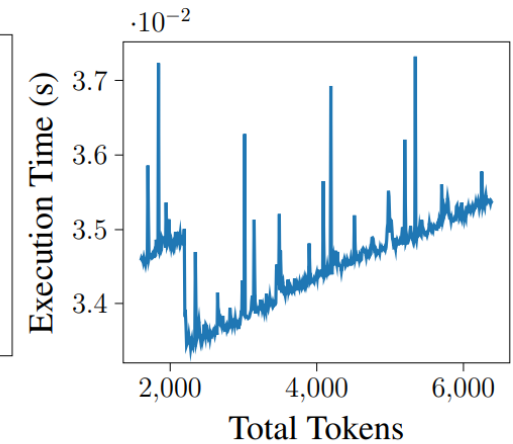
- Where to route
 - Across multiple model instances
 - Question answered by most routing algorithms in use today
- When to route
 - Waiting for the right time, when most information is available
 - This is not done by the LLM routers today

What are the factors that impact the E2E latency?

- Request type
 - Input length
 - Output length



(a) Effect of prompt length



(b) Effect of decode length

- Effect of batching with other requests
 - Longer batch times
 - Preemption due to KV cache out-of-memory
- Queuing Delay

Routing and Batching Algorithm Space

- We test three batching and three routing strategies

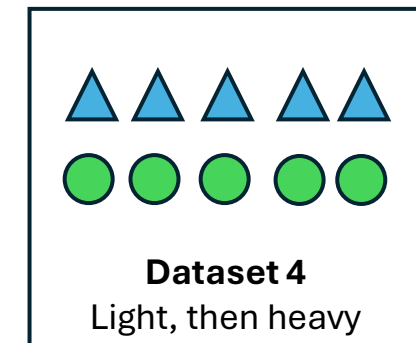
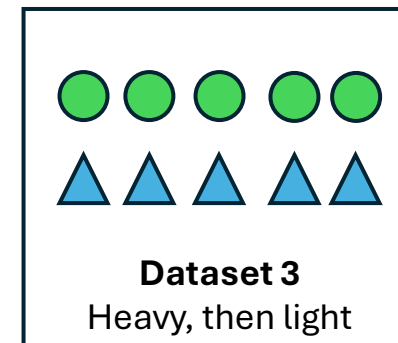
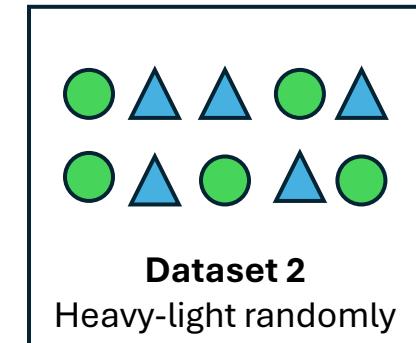
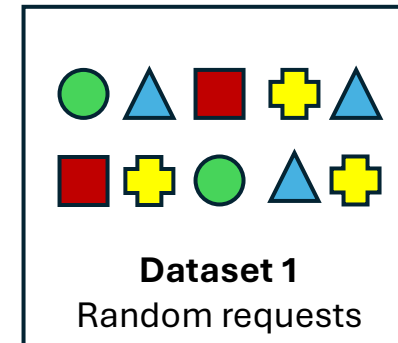
- We use four synthetic datasets for evaluations

Batching Strategies

Firs Come First Serve (FCFS)	Least Work Left (LWL)	Bin Packing
Earliest request gets served first	Request with least decode tokens are prioritized	Requests with most decode tokens are prioritized

Routing Strategies

Dedicated Small Large	Round Robin/ JSQ	Decode Balancers
One model is reserved for large requests	Iterate through models one by one	Balance decode tokens to produce across models



Importance of routing algorithm

Batching Algorithm	Routing Algorithm	Total End to End Latency (seconds)			
		(LH, HL random)	(Random)	(LH, then HL)	(HL, then LH)
Bin Packing [18]	Dedicated Small-Large	704.5	644.75	566.25	588.15
	Round Robin	581.5	559.3	424.8	440.68
	Decode Balancer	595.82	555.4	424.82	440.81
Least Work Left	Dedicated Small-Large	704.5	641.81	566.25	588.15
	Round Robin	585.14	554.00	424.64	440.82
	Decode Balancer	596.95	559.97	424.66	440.81
FCFS [43]	Dedicated Small-Large	704.5	648.66	566.25	588.15
	Round Robin	607.45	572.16	424.80	440.82
	Decode Balancer	605.65	573.17	424.82	440.81

JSQ performs very similar to Round Robin

A good routing algorithm improves upon any batching algorithm and workload mix

What does a router need to know?

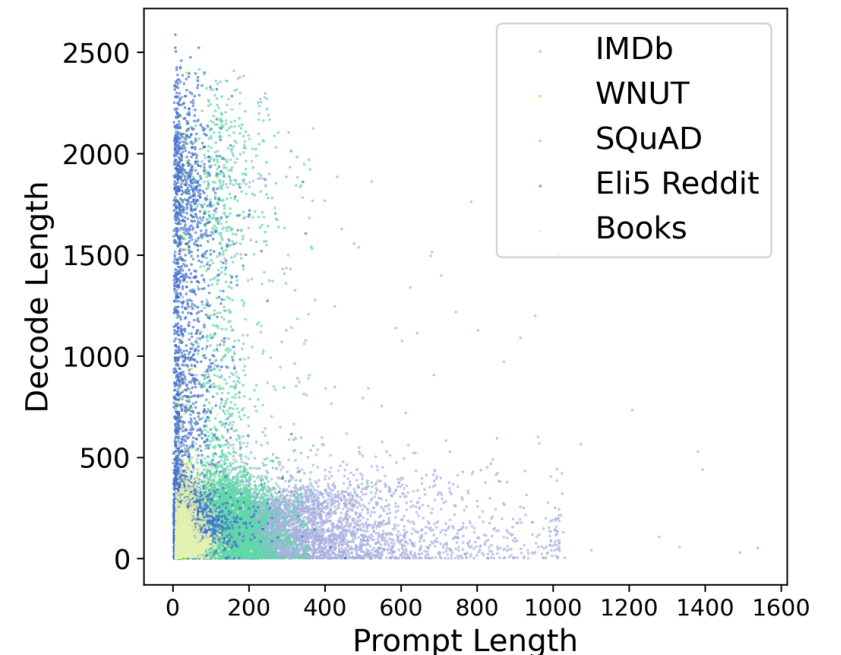
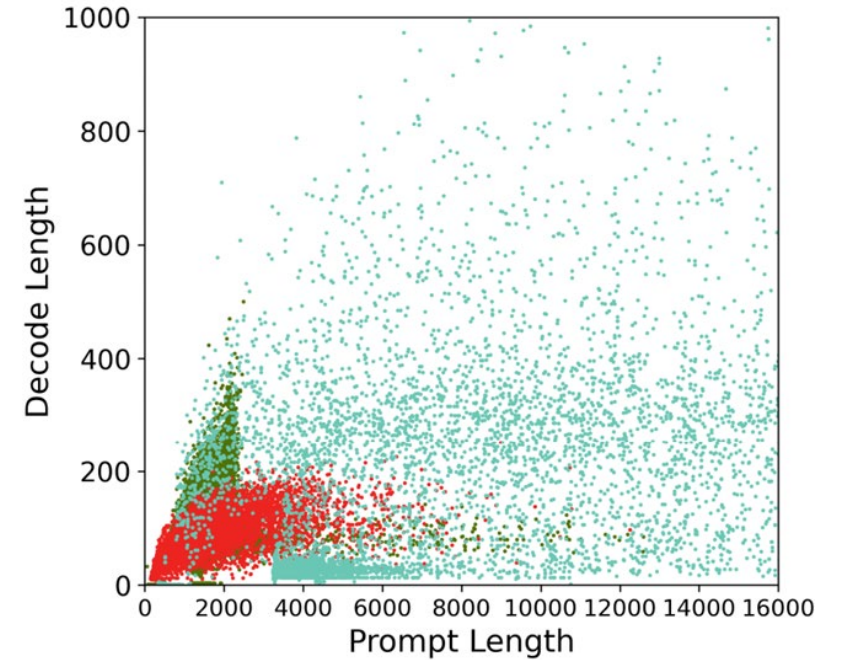
- Current state across the model instances
- Impact of routing the latest request to a given instance
 - Impact depends on the incoming request
 - Impact depends on the types of requests already running in the instance
 - Impact depends on how loaded the model instance already is

Output length of any of these requests is not known while routing or batching

Output length prediction

S³: Increasing GPU Utilization during Generative Inference for Higher Throughput

- ^ Prior work uses output length prediction to size KV allocation
 - DistillBERT fine-tuned
 - Maximum length allowed by the model is divided into equal sized buckets
- We tried five different workloads
 - summarization, sentiment analysis, in-context QnA, no-context QnA, translation
- We find that
 - The distributions are workload-specific
 - The distributions tend to be dense around specific regions
 - Unequal bucket sizes make more sense!



Decode length prediction results

Method	Top-1 Accuracy
Unequal buckets, with tasks (Proposed Approach)	73%
Unequal buckets, without tasks	9.3%
Equal buckets (512 size), with tasks	65%
Equal buckets (512 size), without tasks	5.5%

What does a router need to know?

- Current state across the model instances
- Impact of routing the latest request to a given instance
 - Impact depends on the incoming request
 - Impact depends on the types of requests already running in the instance
 - Impact depends on how loaded the model instance already is



Output length of any of these requests is not known while routing or batching

A workload impact estimator is needed: We use the profiles to compile this analytical estimator (details in the paper)

Router design

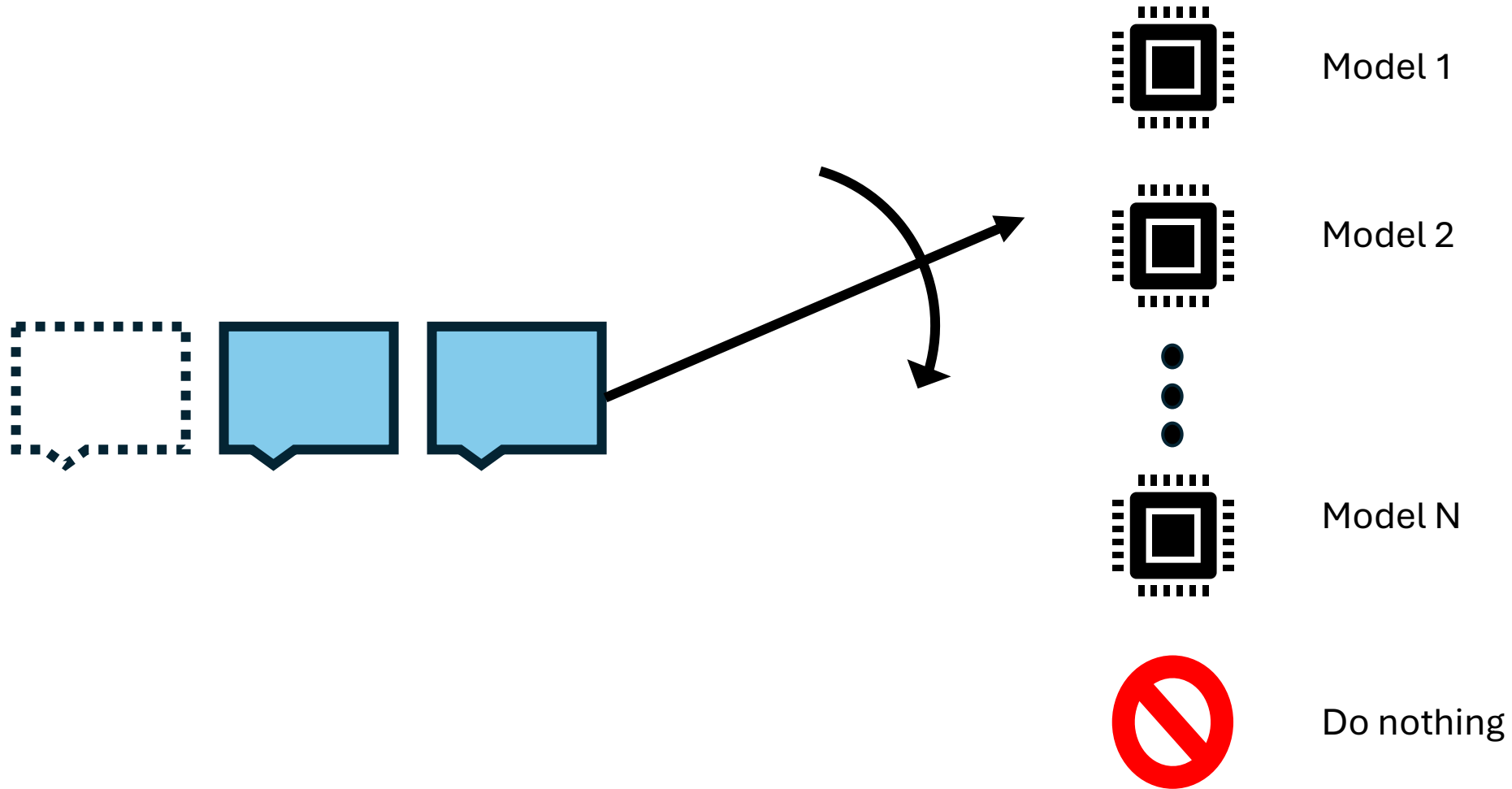
What does a router need to decide?

- Where to route
 - Across multiple model instances
 - Question answered by most routing algorithms in use today
- When to route
 - Waiting for the right time, when most information is available
 - This is not done by the LLM routers today

We propose lightweight Reinforcement Learning (RL)

- LLM workloads have distinct prefill (prompt) and decode phases, which have different compute and memory demands.
- Mixing requests with diverse characteristics (e.g., heavy prompt vs. heavy decode) at the same instance can cause latency spikes.
- Assigning the right request to the right instance at the right time is critical.
- RL is a good fit here because the problem is sequential, dynamic, and stateful—the system learns from feedback (latency, queue times, etc.) over time.

RL Formulation – Action Space



RL Formulation

- A discrete-time Markov Decision Process (MDP):

$$M = (S, A, P, r, \gamma)$$

States S: At time t , state includes:



- Number of requests in the queue w_{qt}
- New prompt length p_t and **estimated decode length bucket** d_t
- Matrices P_t, D_t distributions of prompt and decode token lengths across model instances
- Model instance capacities C_t
- Estimated completion time of the earliest request per instance T_{ct}

RL Formulation

Actions A: Choose one of m model instances to assign the incoming request to, or delay assignment.

Transition P: Determined by the system dynamics as requests are routed and completed.

Reward: Reward function balances:




1.  **Penalty** for queueing delays
2.  **Reward** for completing a request

Workload (heuristic)-guided RL Formulation

Actions A: Choose one of m model instances to assign the incoming request to, or delay assignment.

Transition P: Determined by the system dynamics as requests are routed and completed.

Reward: Reward function balances:

1.  **Penalty** for queueing delays
2.  **Reward** for completing a request
3.  **Penalty for bad workload mixing**, based on a learned workload impact estimator (latency spike model)

Final reward formulation

$$r_t = \underbrace{- \sum_j \frac{1}{T_j} (1 - f_{jt})}_{\text{queue penalty}} + \underbrace{\sum_{j=1}^m \sum_i r_w \cdot w_{mit}}_{\text{reward for completion}} - \underbrace{(\gamma - \gamma_k) h(s_t, s_{t+1})}_{\text{heuristic-guided penalty}}$$

The **mixing cost heuristic** penalizes routing decisions that cause bad mixing (leading to latency spikes). We slowly decay it.

Without guidance: RL can flail around for a long time before discovering good routing strategies.

With guidance: The agent starts off making "reasonable" decisions, guided by known good behaviors, then gradually finds better ones.

It's like teaching someone chess by letting them follow book openings at first — but eventually they figure out new tactics on their own.

We try three variants of RL

$$r_t = \underbrace{-\sum_j \frac{1}{T_j}(1 - f_{jt})}_{\text{queue penalty}} + \underbrace{\sum_{j=1}^m \sum_i r_w \cdot w_{mit}}_{\text{reward for completion}} - \underbrace{(\gamma - \gamma_k)h(s_t, s_{t+1})}_{\text{heuristic-guided penalty}}$$

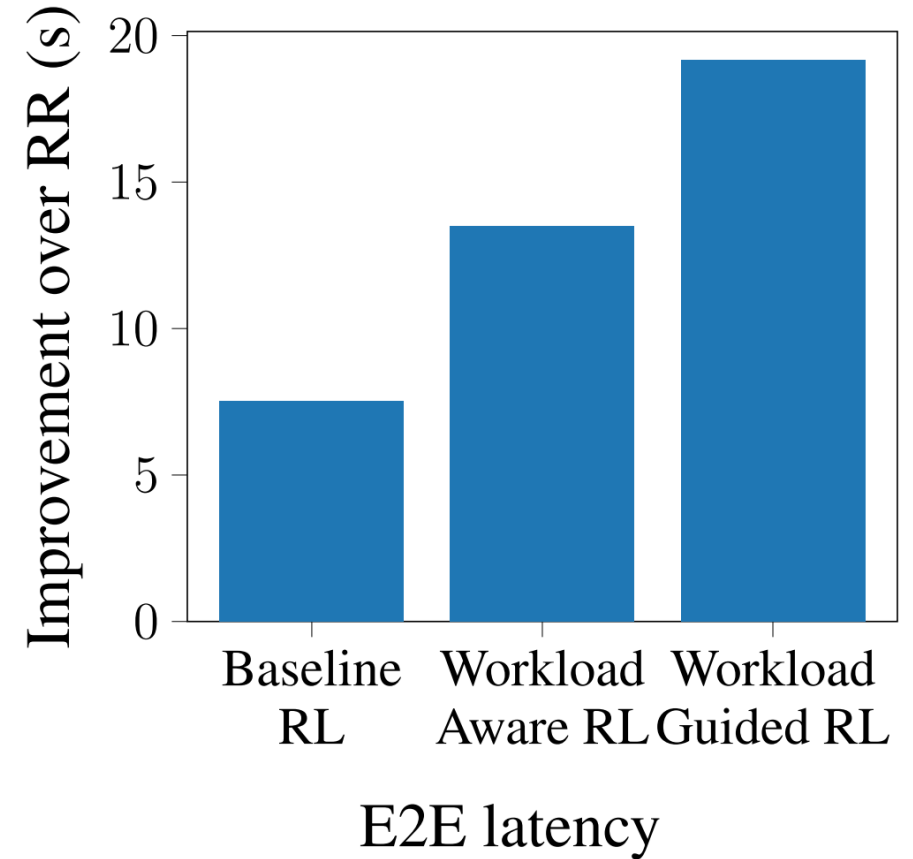
- Baseline RL: Without the heuristic
- Workload-aware RL: With the heuristic, without the decay
- Workload-guided RL: With the heuristic gradually decaying

Results – End-to-end latency

- Servicing 2000 requests with 4 model instances (V100, Llama-2-7b)
- Using Round Robin as benchmark

Improvements

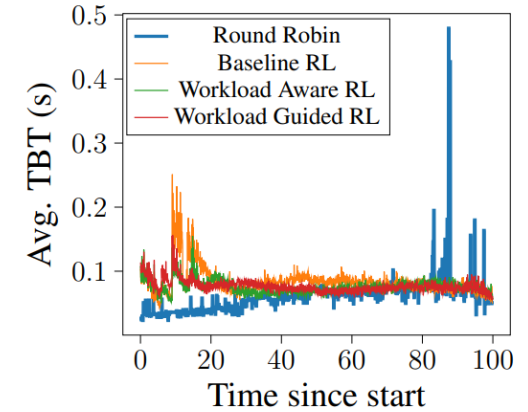
- Baseline RL 7.54 seconds (4.35%)
- Workload Aware RL 13.50 seconds (7.79%)
- Workload Guided RL 19.18 seconds (11.43%)



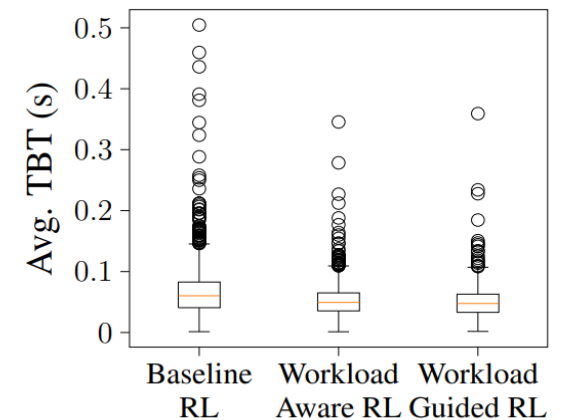
Results – Time Between Tokens (TBT)

- Calculated after first token is generated
- Distribution plot shows lesser variance in average TBT

TBT no longer spikes because of incoming prompts!

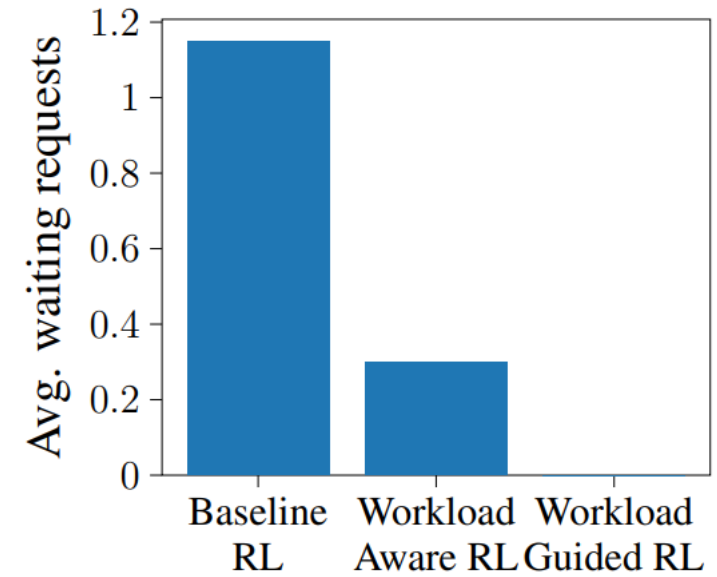
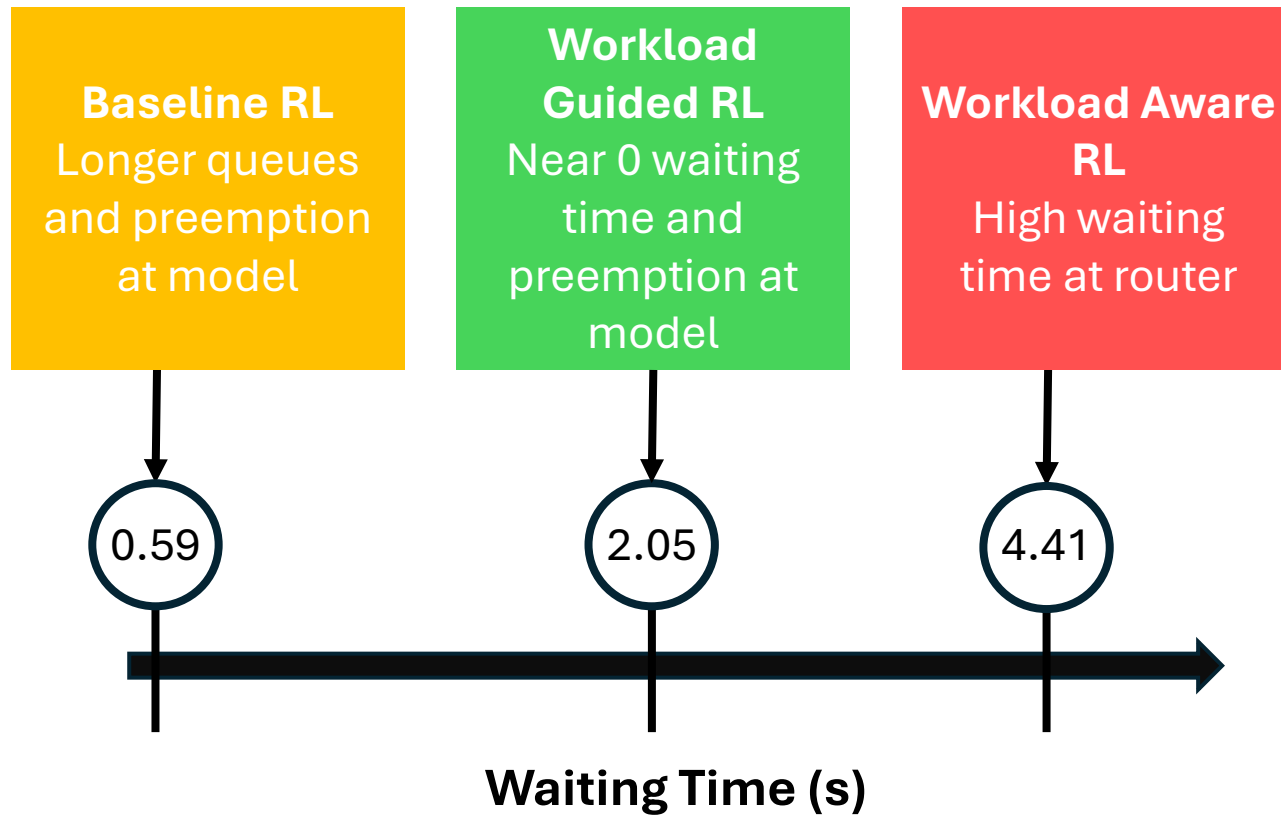


Average TBT of requests served



Average TBT distribution

Results – Waiting Time at Router and Model



(c) Queue length at model instance

Results – Generalizability and Scalability

Our method is generalizable to different hardware, prefill chunking and more instances!

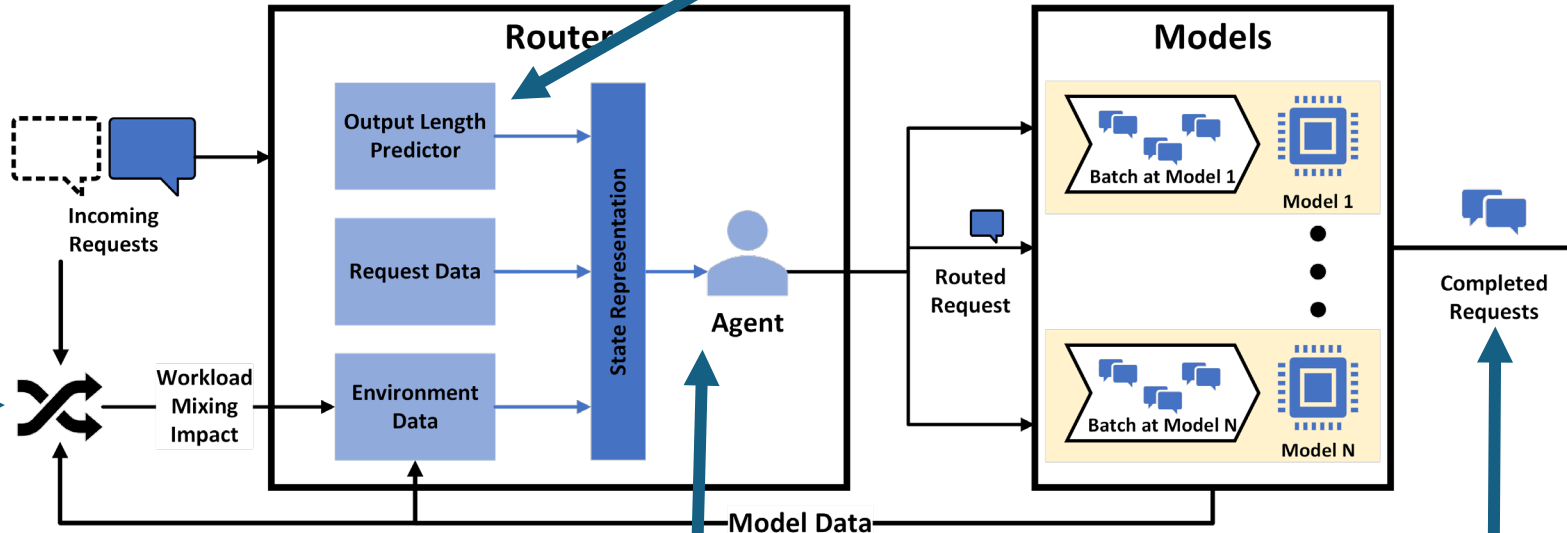
Routing Algorithm	Prefill Chunking	Avg. E2E Latency (s)	Improvement
Round Robin	No	248.41	-
Baseline RL	No	240.58	3.15%
Workload Aware RL	No	231.66	6.74%
Workload Guided RL	No	221.80	10.71%
Round Robin	Yes	247.30	0.45%
Baseline RL	Yes	240.68	3.11%
Workload Aware RL	Yes	231.12	6.96%
Workload Guided RL	Yes	220.93	11.06%

Table 3. Intelligent router was able to generalize the approach across different model and hardware combinations, outperforms heuristics, and shows additional improvements even with chunked prefills.

Highlights

Fixed shortcomings of SOTA.
Improved performance
from ~8% to 73% !

First of a kind
study on impact of
mixing workloads



RL based router with the aim
to minimize E2E latency. Learning
guided with workload mixing impact

Upto 11% reduction in
average E2E
latency over current
methods!

Questions?

Experimental Details – Infrastructure

- 4 model instances, each managed with vLLM (FCFS scheduler)
- Hardware: V100
- Model: Llama 2 7B
- Datasets: Books (translation), IMDb (sentiment analysis), SQuAD (in-context QnA), Elli5 Reddit (no-context QnA), WNUT (entity recognition)
- 2000 requests

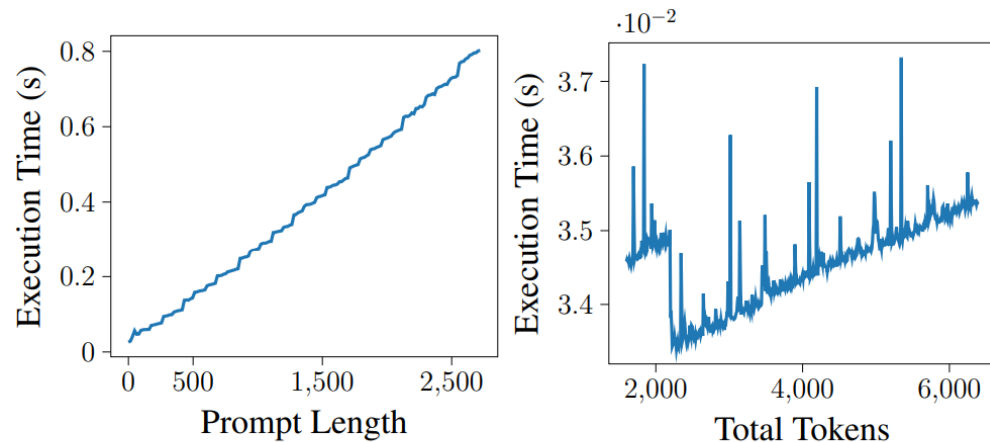
Experimental Details – Hyperparameters

- Arrival rate: 20 requests per second
- Action interval: 0.02 seconds (minimum decode iteration time)
- Exploration factor: decays within episode from 0.99 for first 30 episodes (decay factor 0.5)

Workload Impact Estimator

Latency due to prefill and decode tokens increases linearly

Calculate impact of mixing as linear combination of two



(a) Effect of prompt length

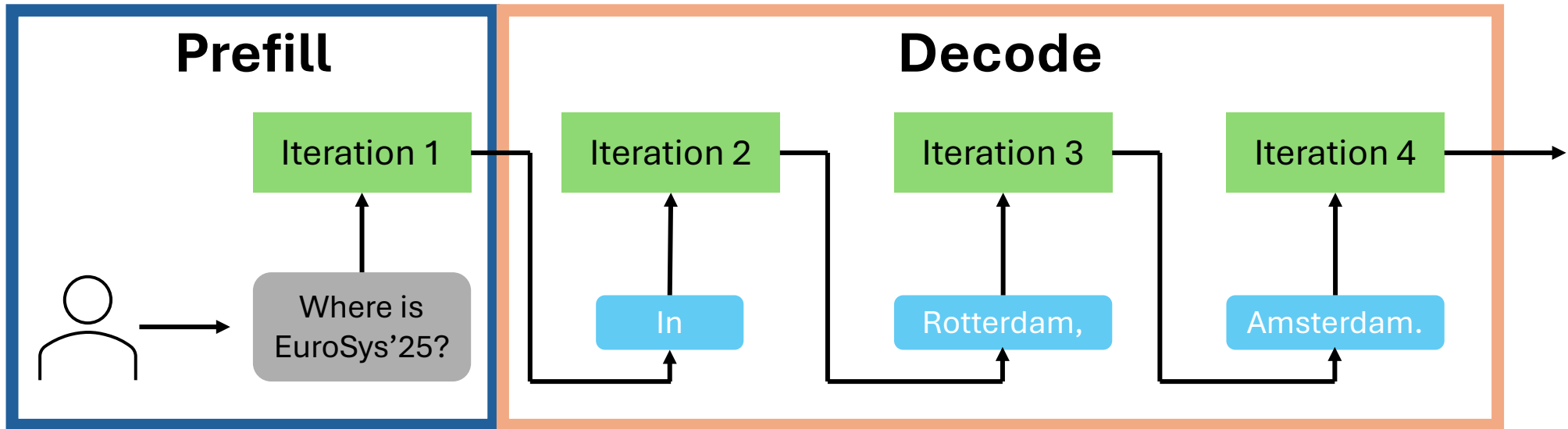
(b) Effect of decode length

$$T_{p_i}^m = \text{grad}_1 * \left((p_i^2 + \sum_{j=1}^n (p_j^m + d_j^m)) \right)$$

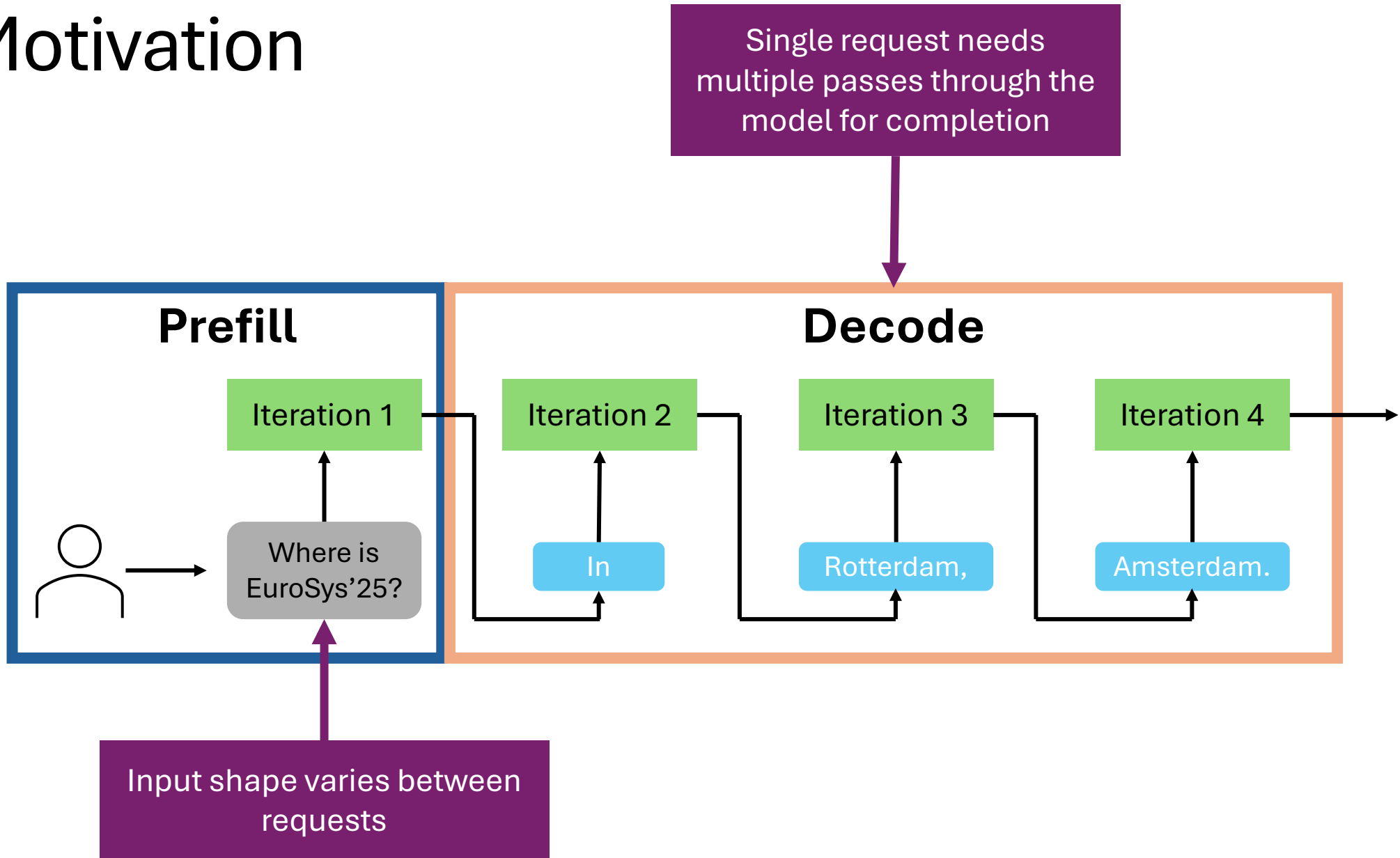
$$r_{p_i}^m = \begin{cases} 1 & \text{if } T_{p_i}^m \leq \epsilon \\ 1 - \frac{T_{p_i}^m}{\epsilon} & \text{otherwise} \end{cases}$$

$$r_d^m = -\text{grad}_2 * \sum_{j=1}^n (p_j^m + d_j^m) + p_i + d_i$$

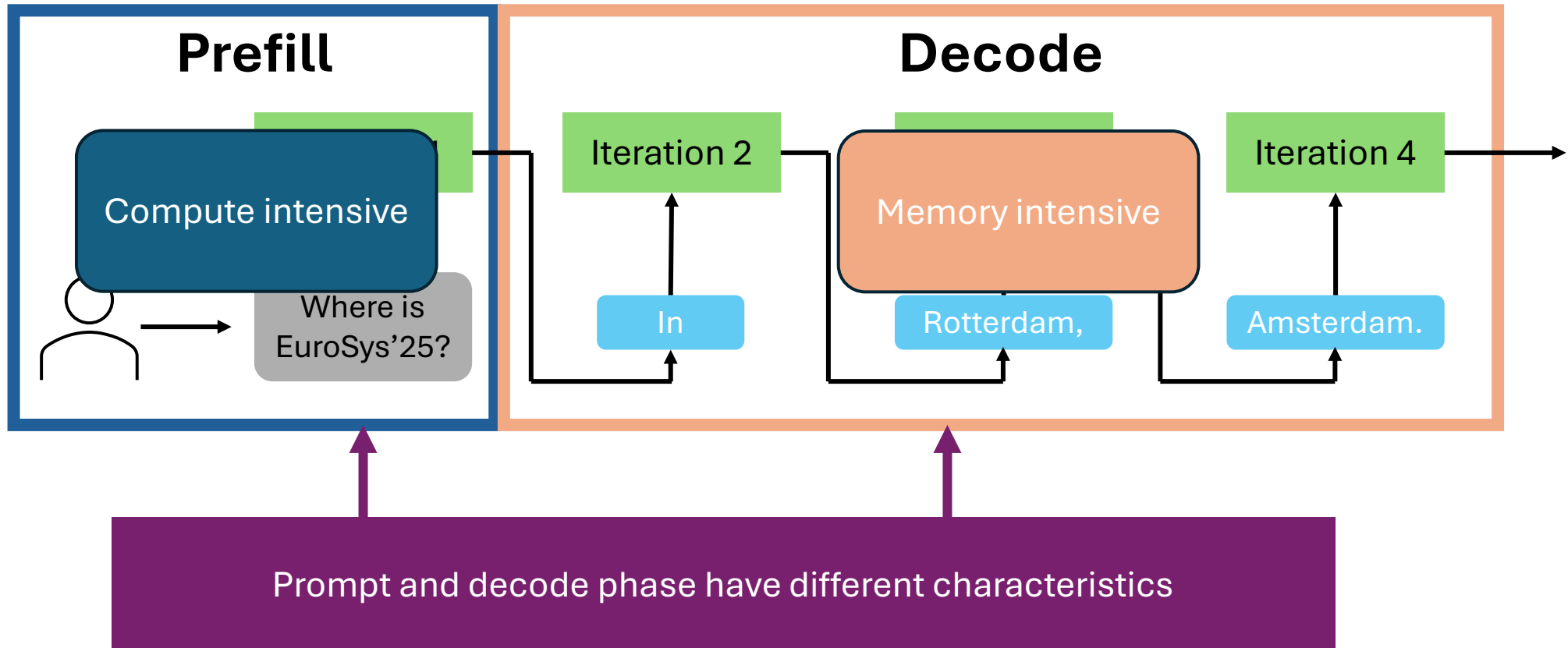
Motivation



Motivation



Motivation



Motivation

