Splitwise

Efficient Generative LLM Inference Using Phase Splitting

Pratyush Patel

Esha Choukse Chaojie Zhang Aashaka Shah Íñigo Goiri
Saeed Maleki Ricardo Bianchini


ChatGPT
Gemini
GitHub
Copilot

ChatGPT Gemini GitHub Copilot

Microsoft places huge cap-ex bets on datacenters for cloud and AI

CFO

Google Cloud braces for AI compute costs, ramps up data center inve

Zuckerberg's Meta Is Spending Billions to Buy 350,000 Nvidia H100 GPUs

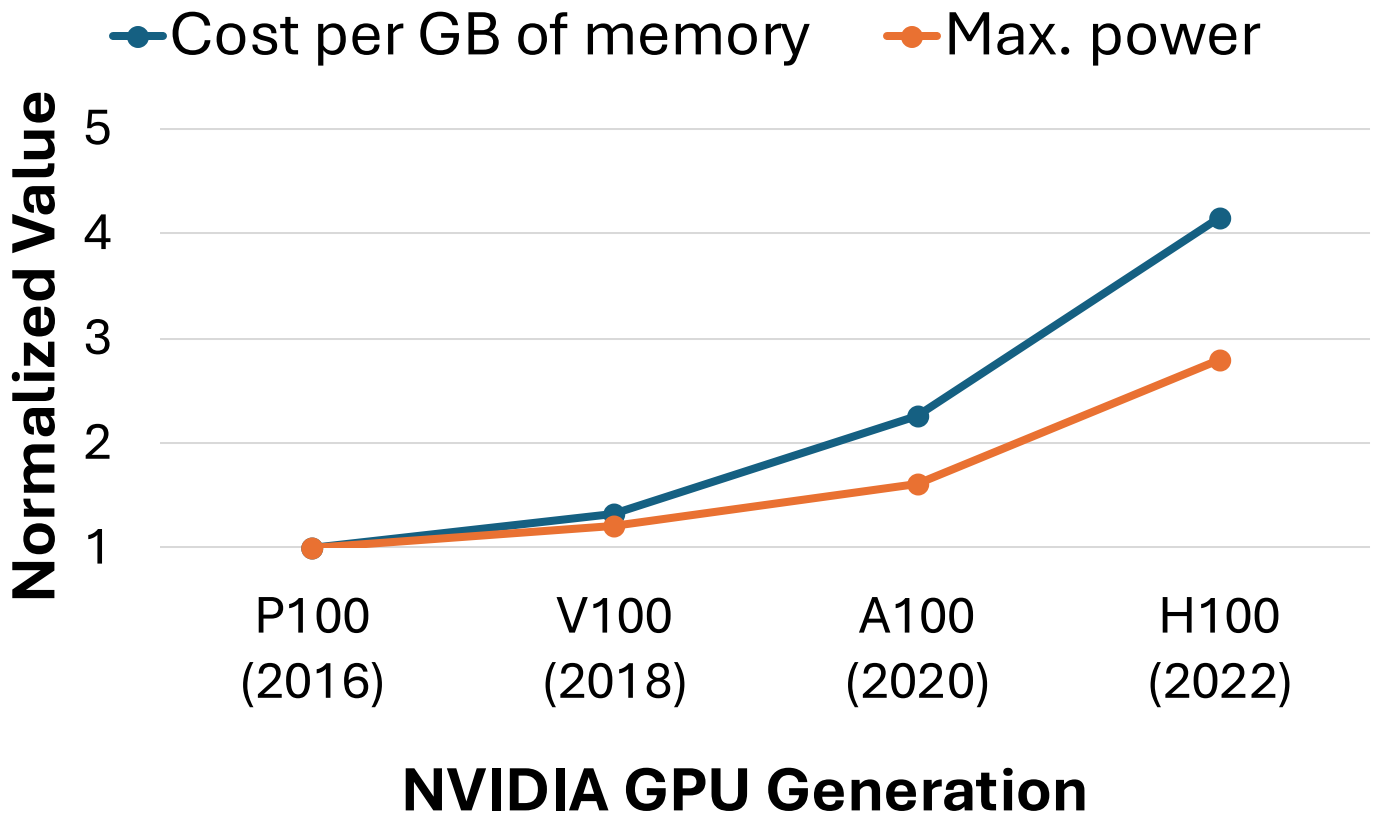
In total, Meta will have the compute power equivalent to 600,000 Nvidia H100 GPUs to help it develop next-generation AI, says CEO Mark Zuckerberg.



ChatGPT
Gemini

GitHub Copilot

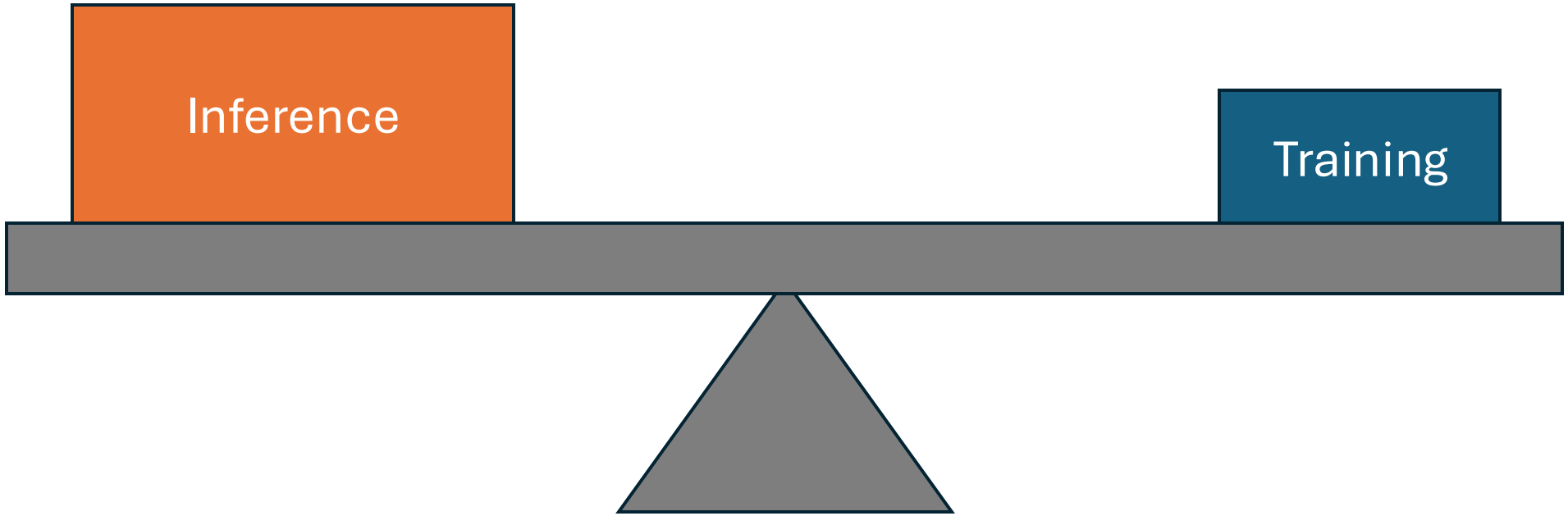

LLM clusters are very expensive and power hungry



https://cloud-gpus.com/

ChatGPT Gemini GitHub Copilot

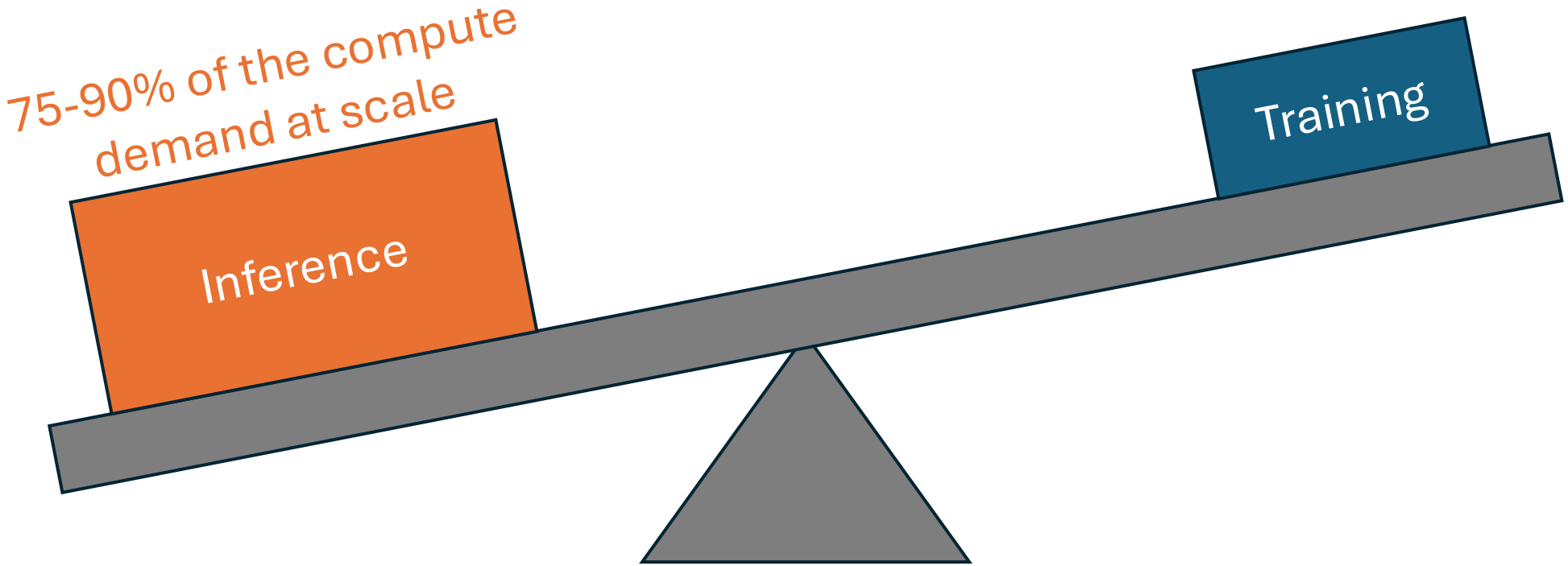
Inference demand far outweighs that of training



Sustainable AI: Environmental Implications, Challenges and Opportunities, MLSys'22

ChatGPT Gemini GitHub Copilot

Inference demand far outweighs that of training



Sustainable AI: Environmental Implications, Challenges and Opportunities, MLSys'22

Splitwise optimizes LLM serving at scale

Characterize generative LLM inference and identify distinct prompt and token phases

Split inference onto different servers for phase-specific resource management

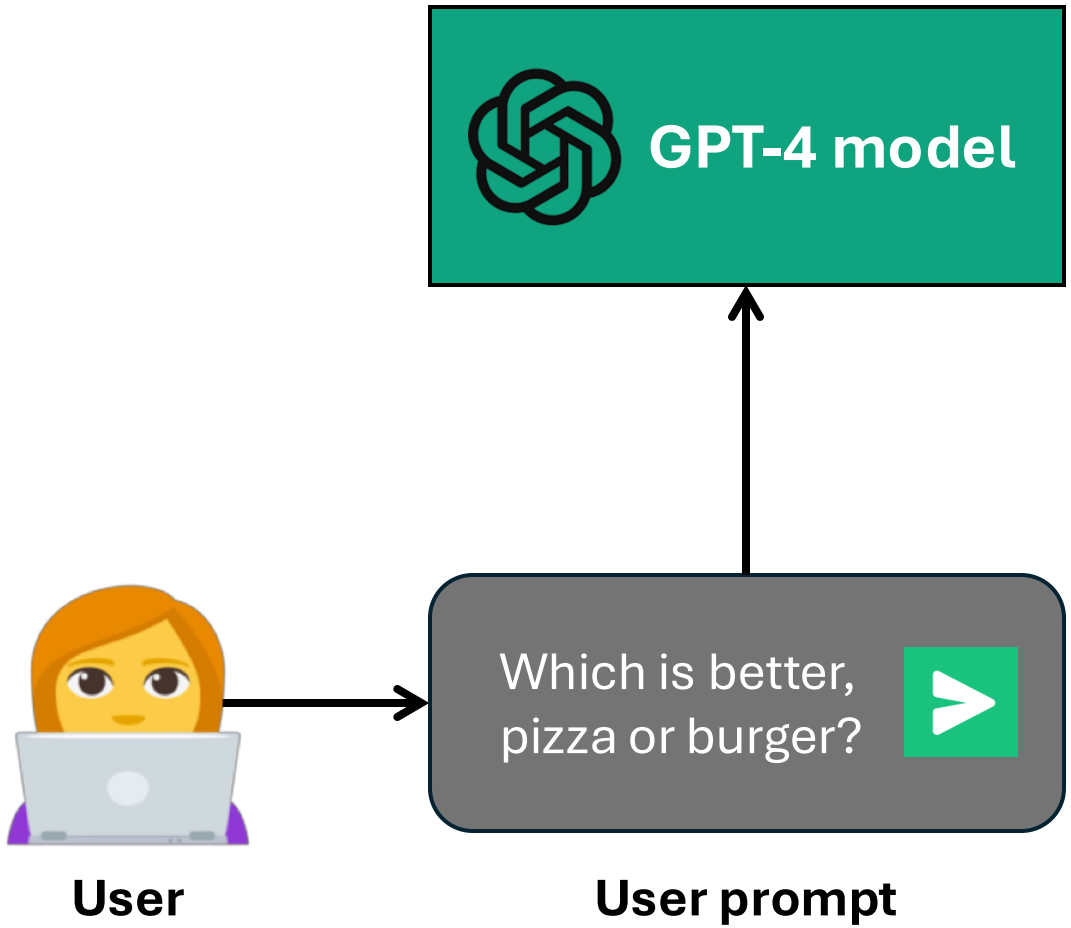
Design clusters using Splitwise, which improves efficiency across various metrics

Anatomy of a generative LLM inference

ChatGPT

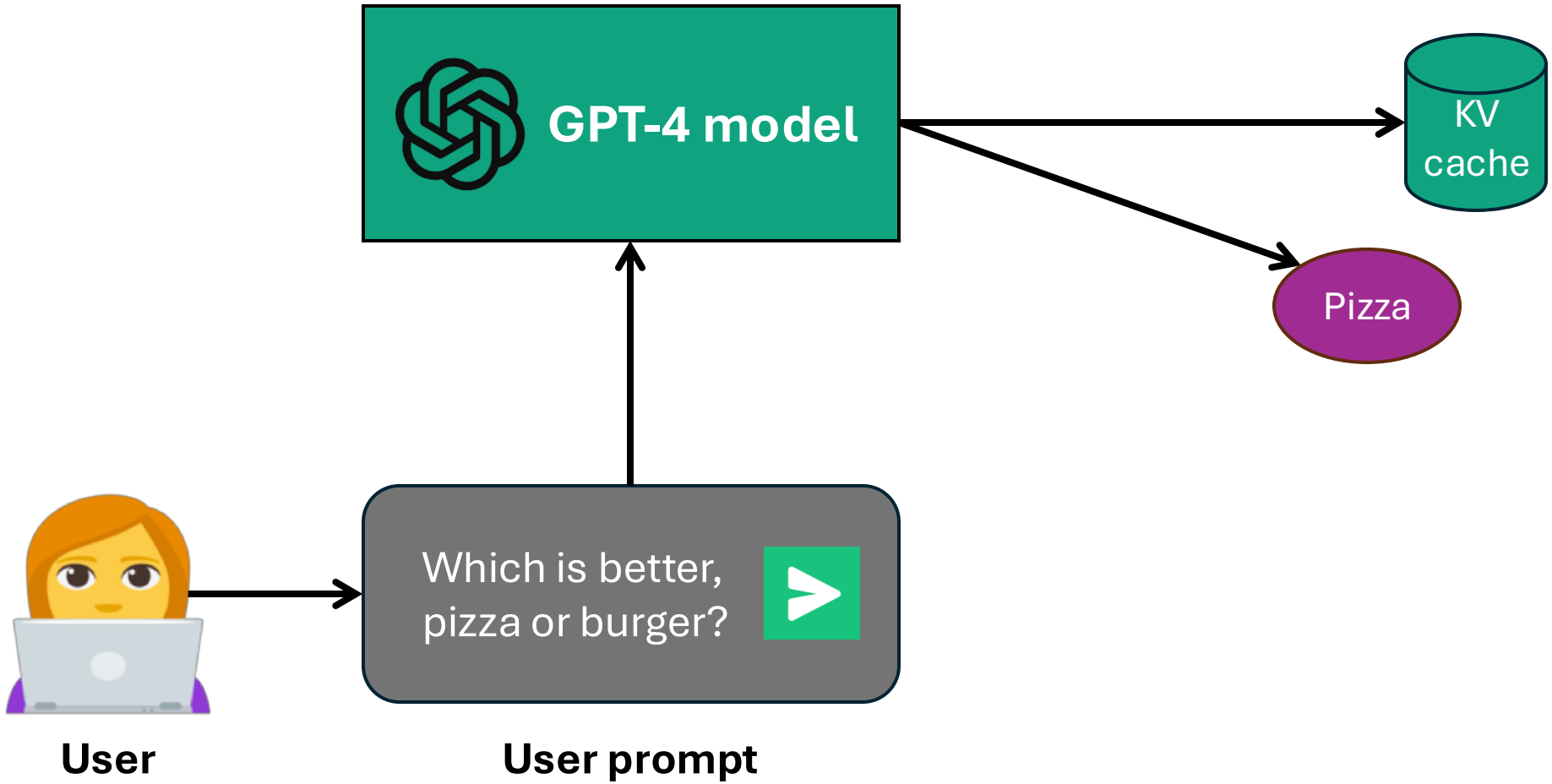
Anatomy of a generative LLM inference

User submits a prompt to the LLM



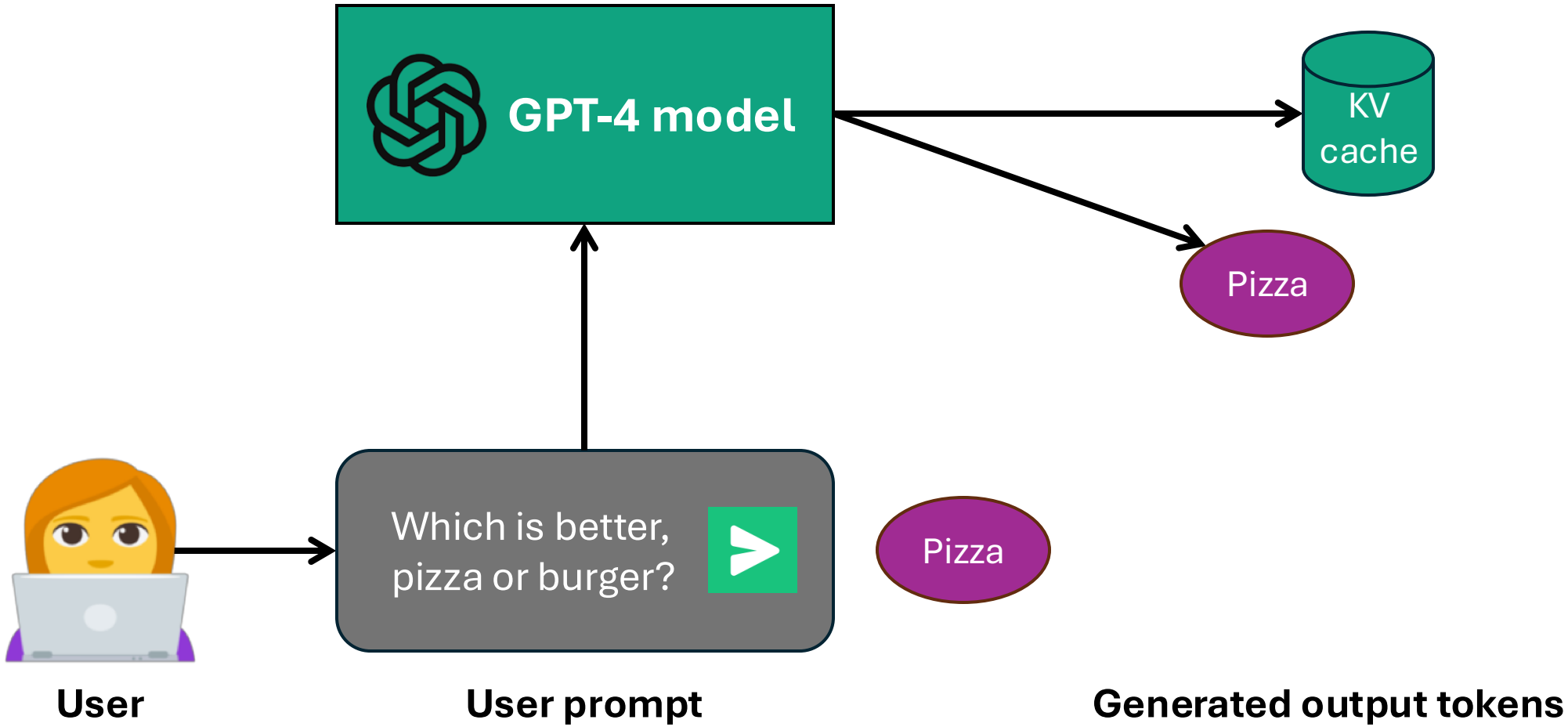
Forward pass 1:

LLM processes the prompt to generate first output token



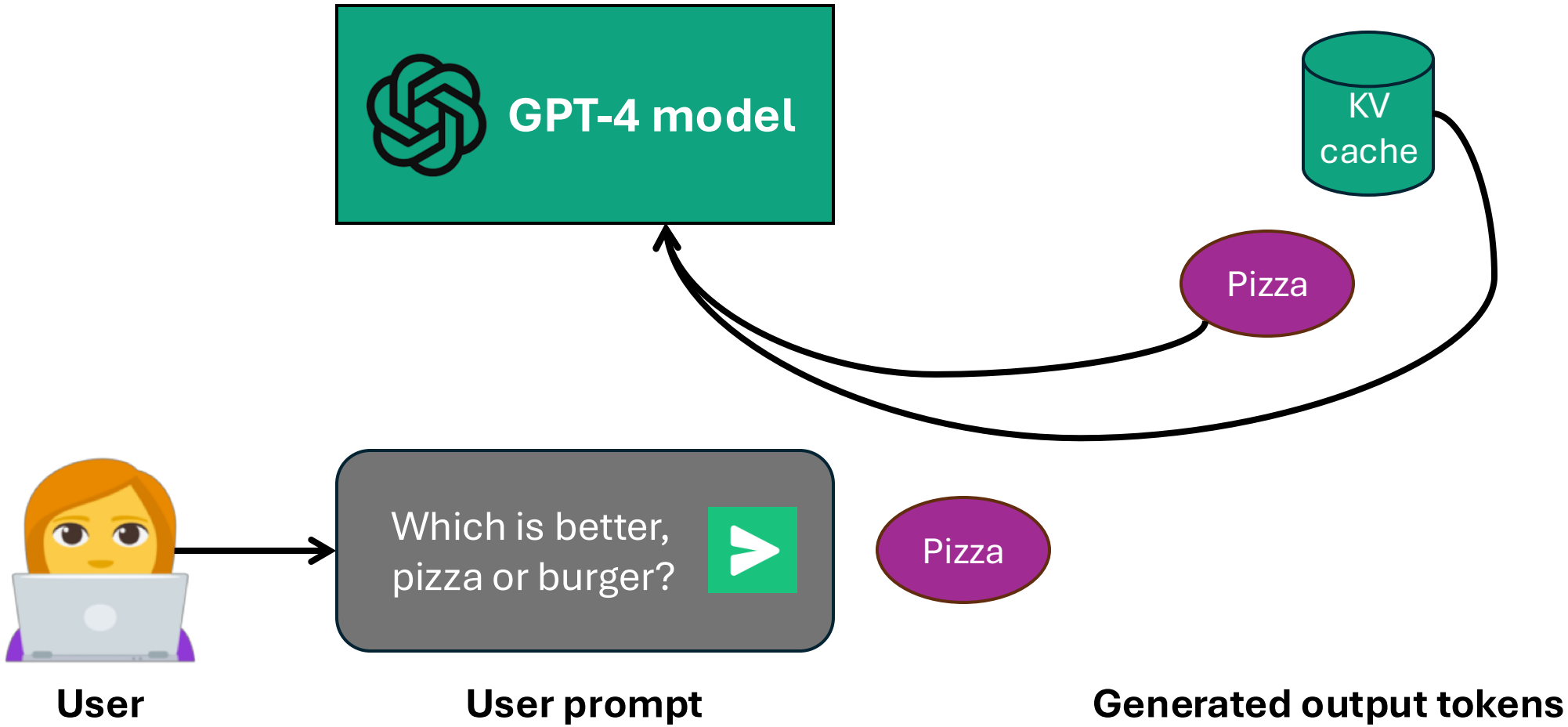
Forward pass 1:

LLM processes the prompt to generate first output token



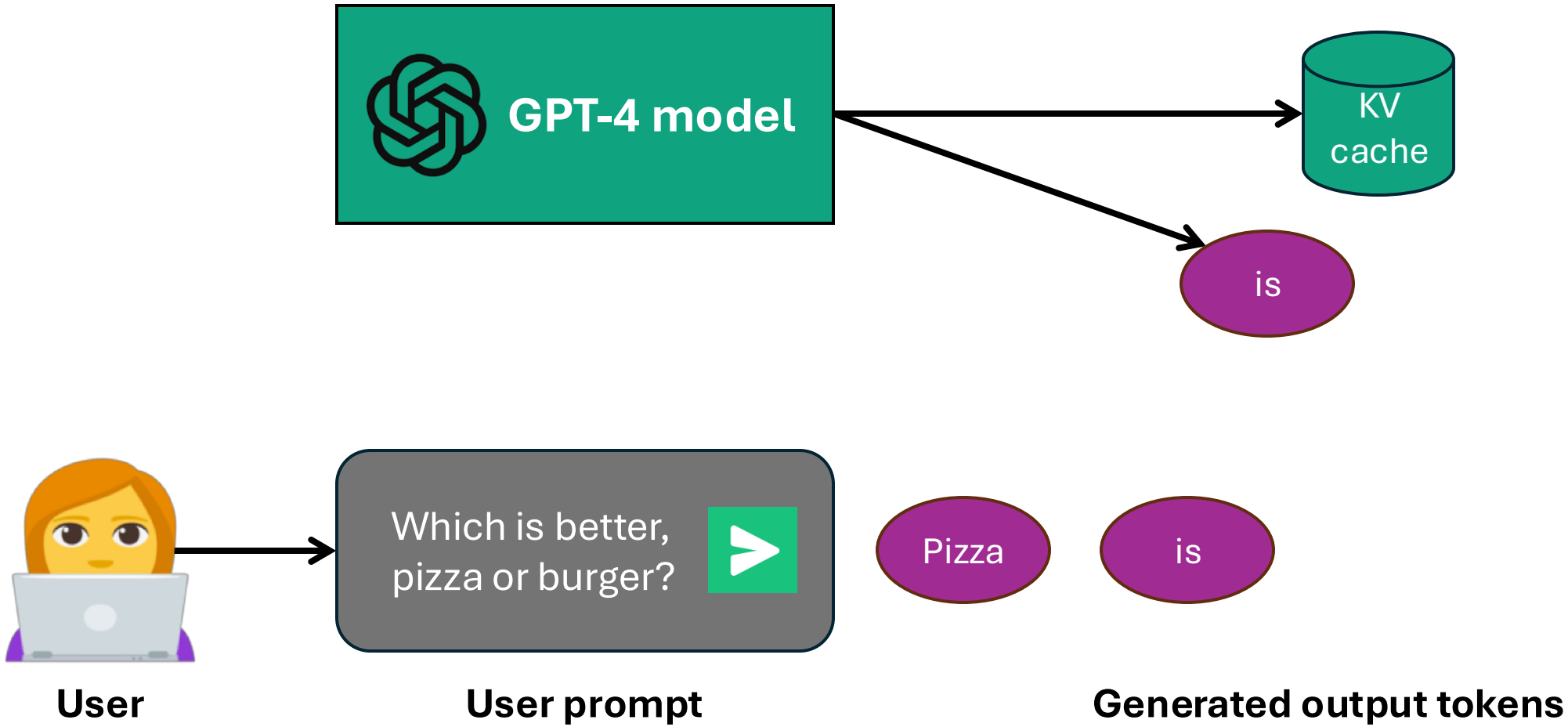
Forward pass 2:

LLM generates next token using KV cache and previous token



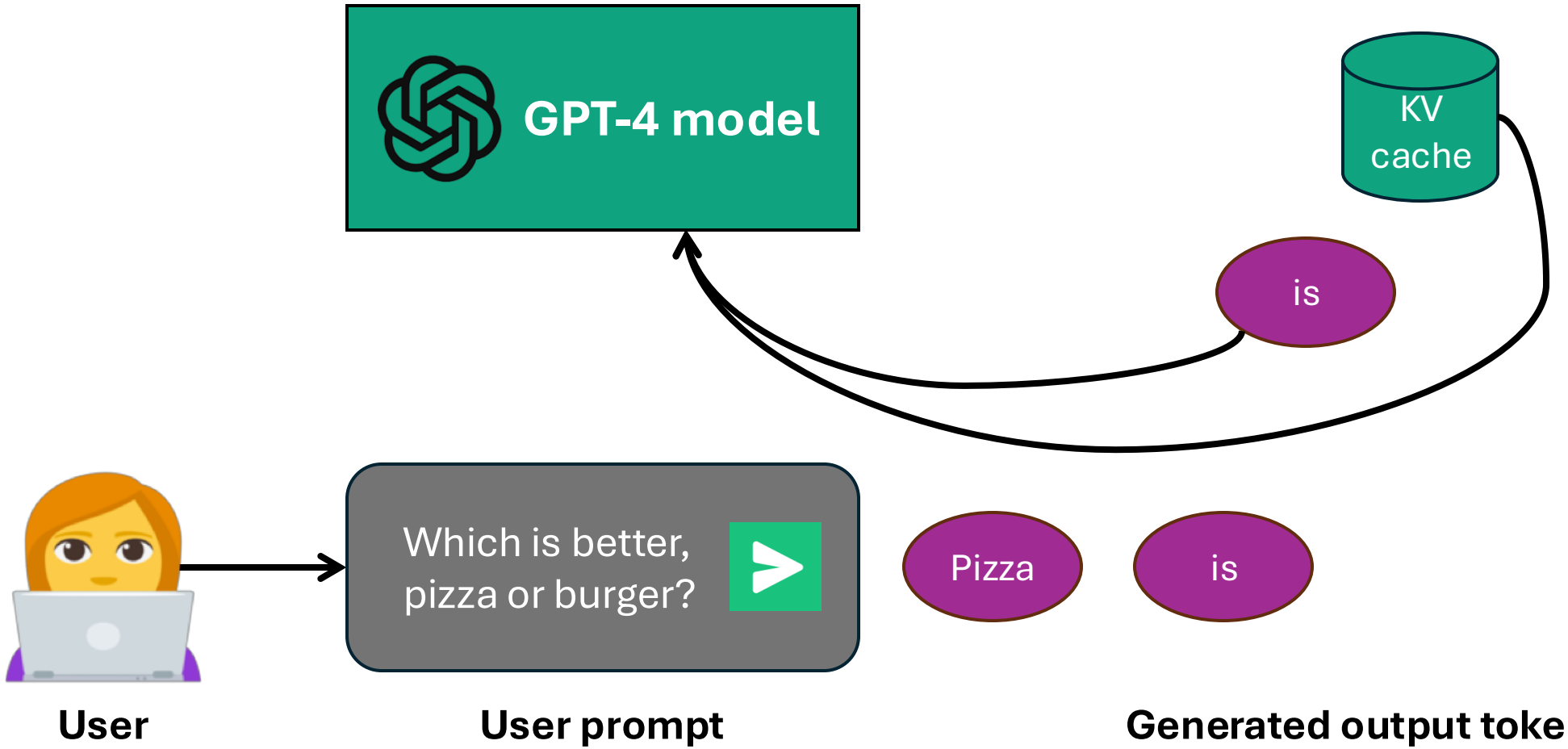
Forward pass 2:

LLM generates next token using KV cache and previous token



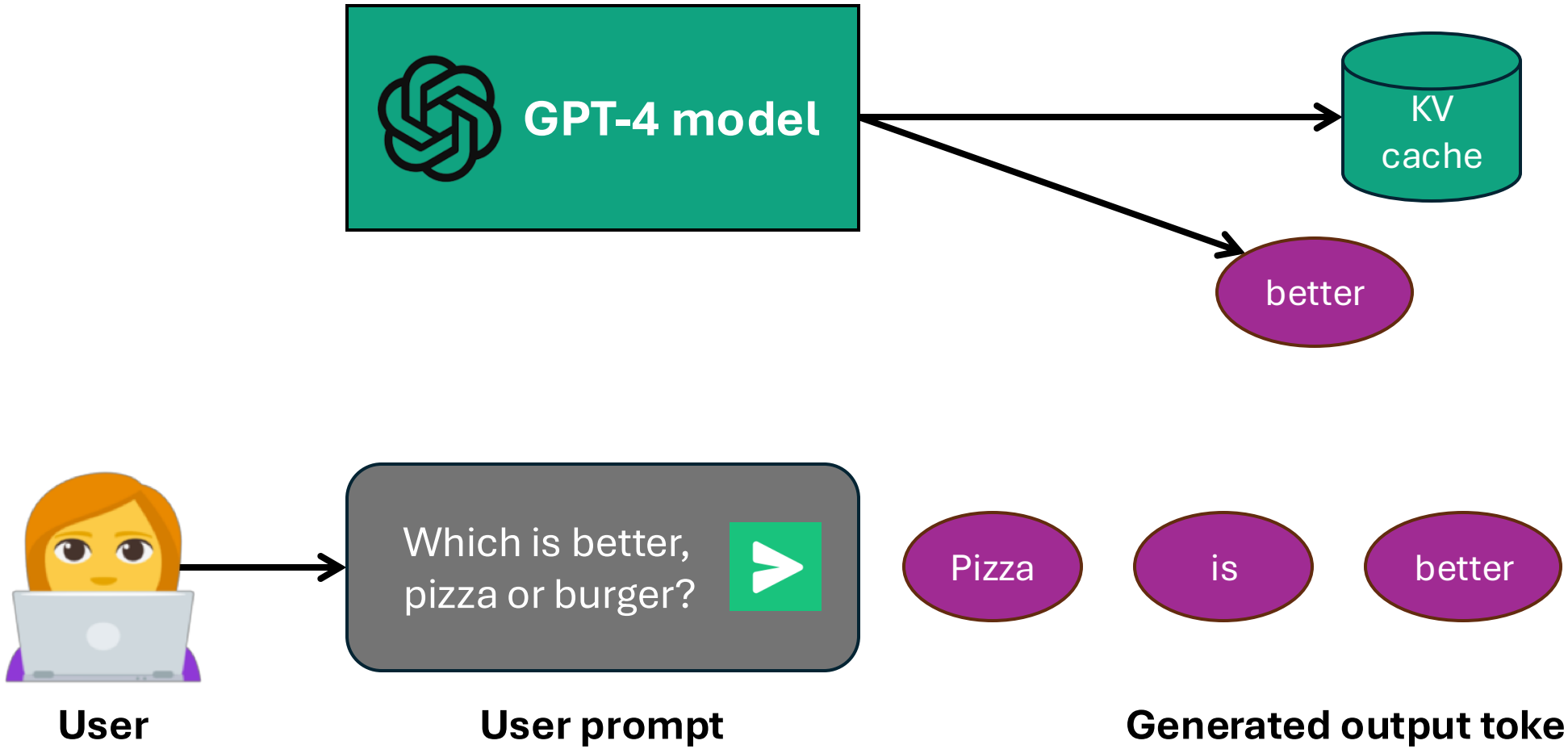
Forward pass 3:

LLM generates next token using KV cache and previous token



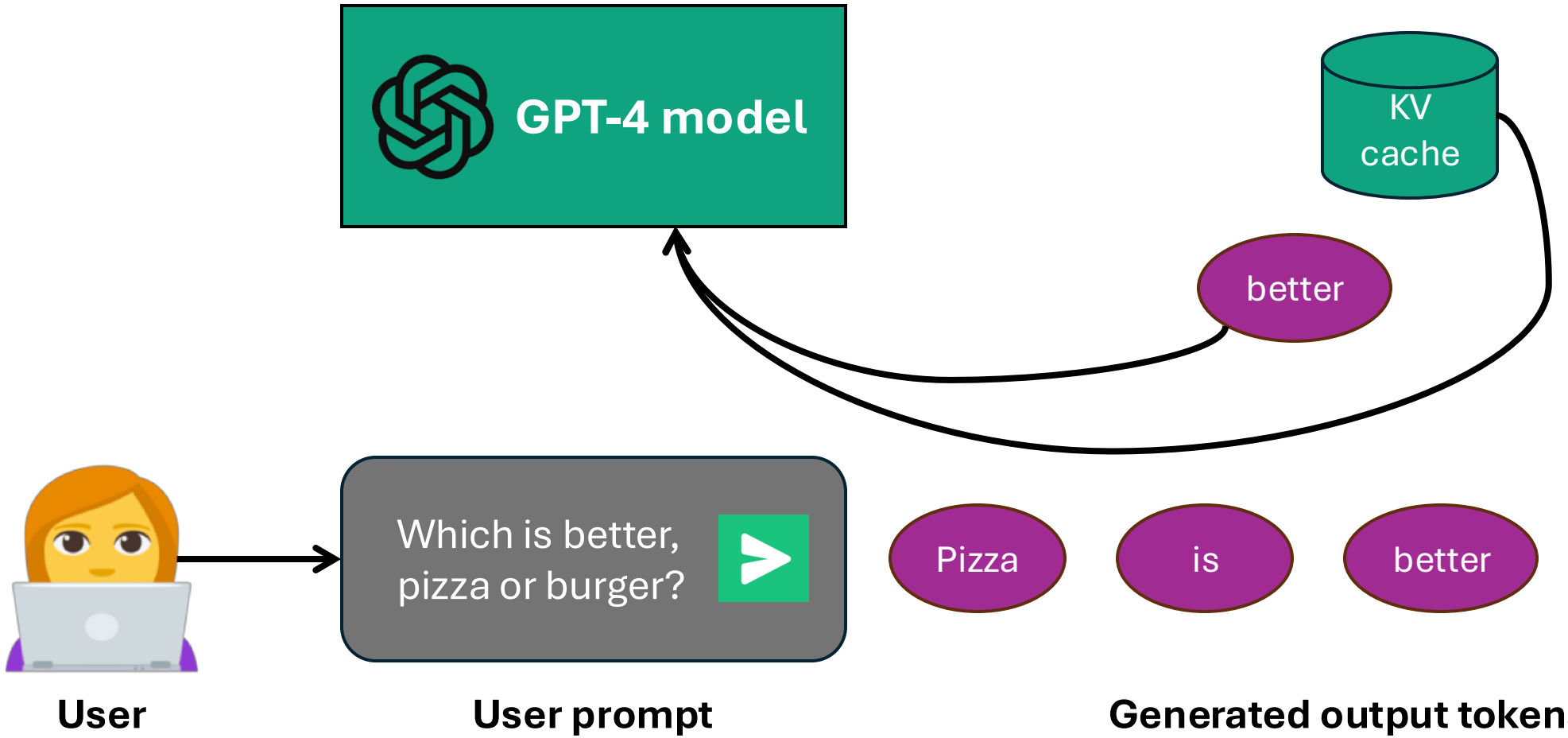
Forward pass 3:

LLM generates next token using KV cache and previous token



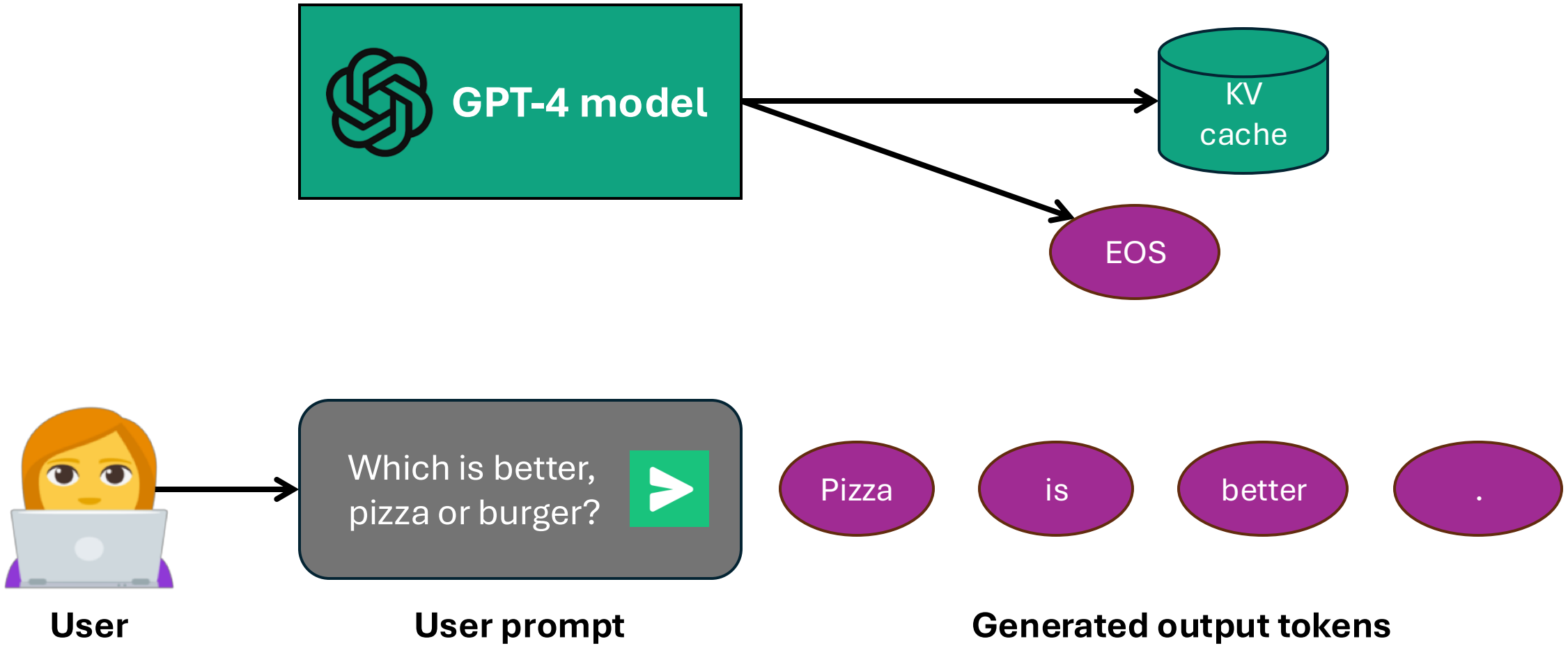
Forward pass 4:

LLM generates next token using KV cache and previous token

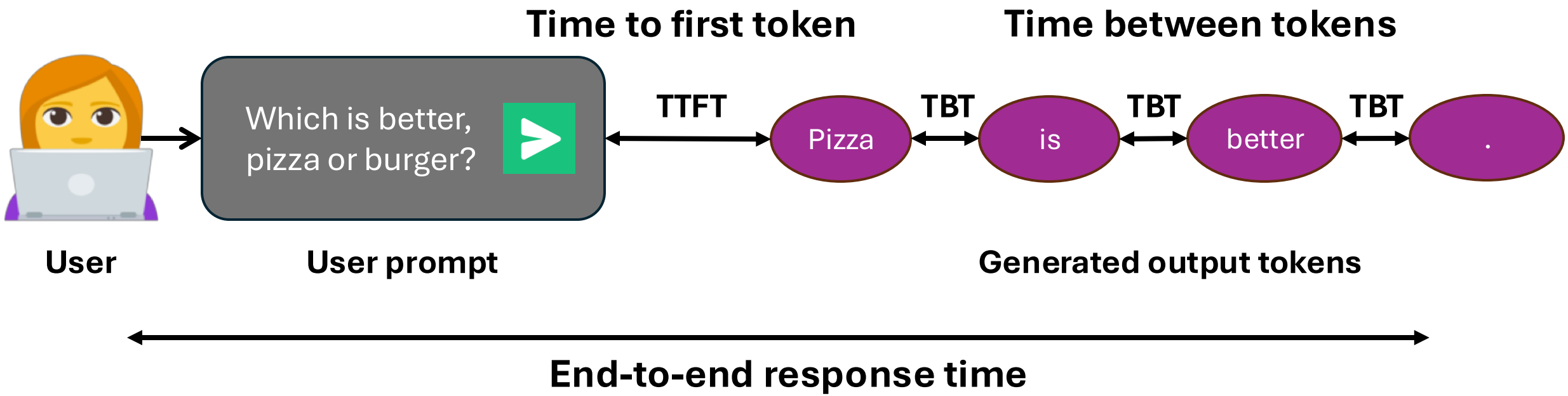


Forward pass 4:

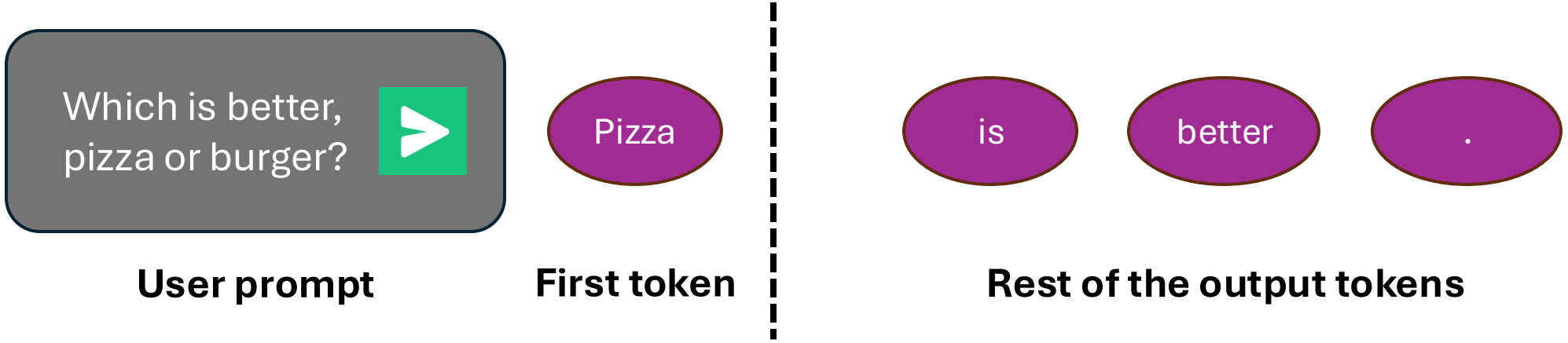
LLM generates next token using KV cache and previous token



Latency metrics for LLM inference

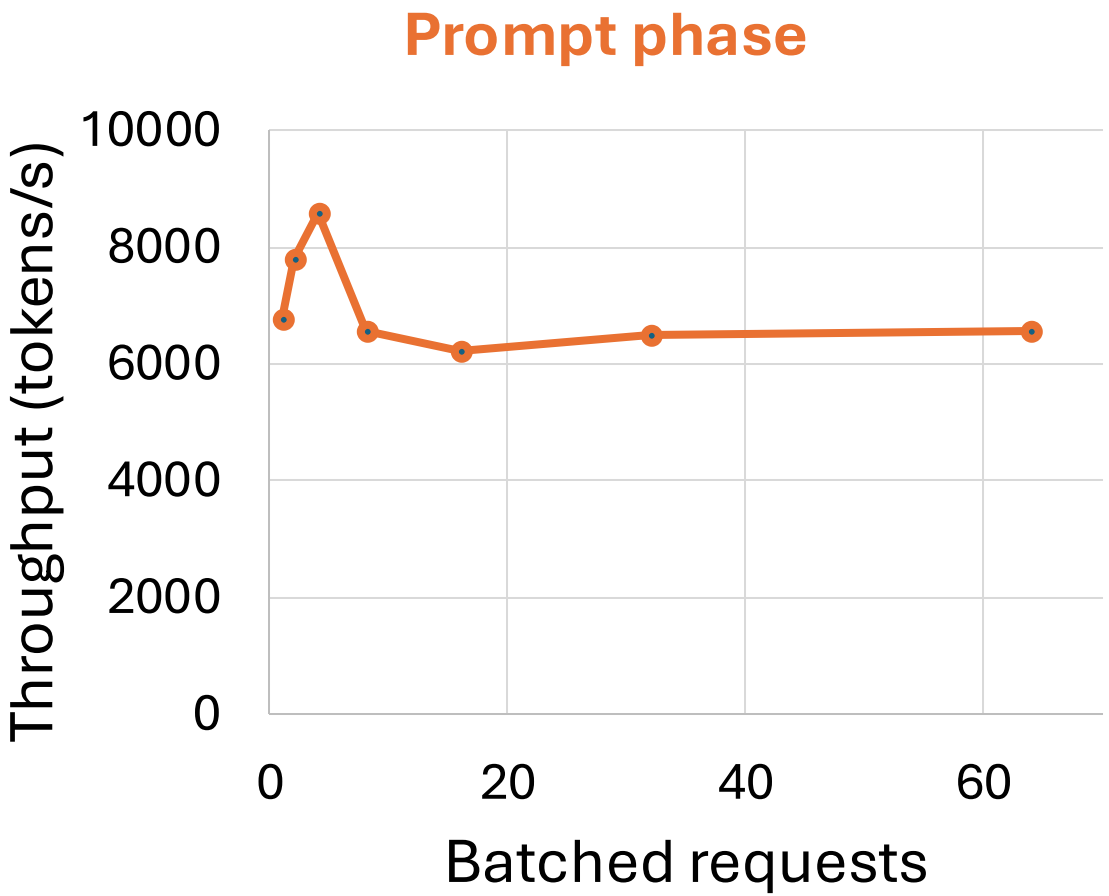


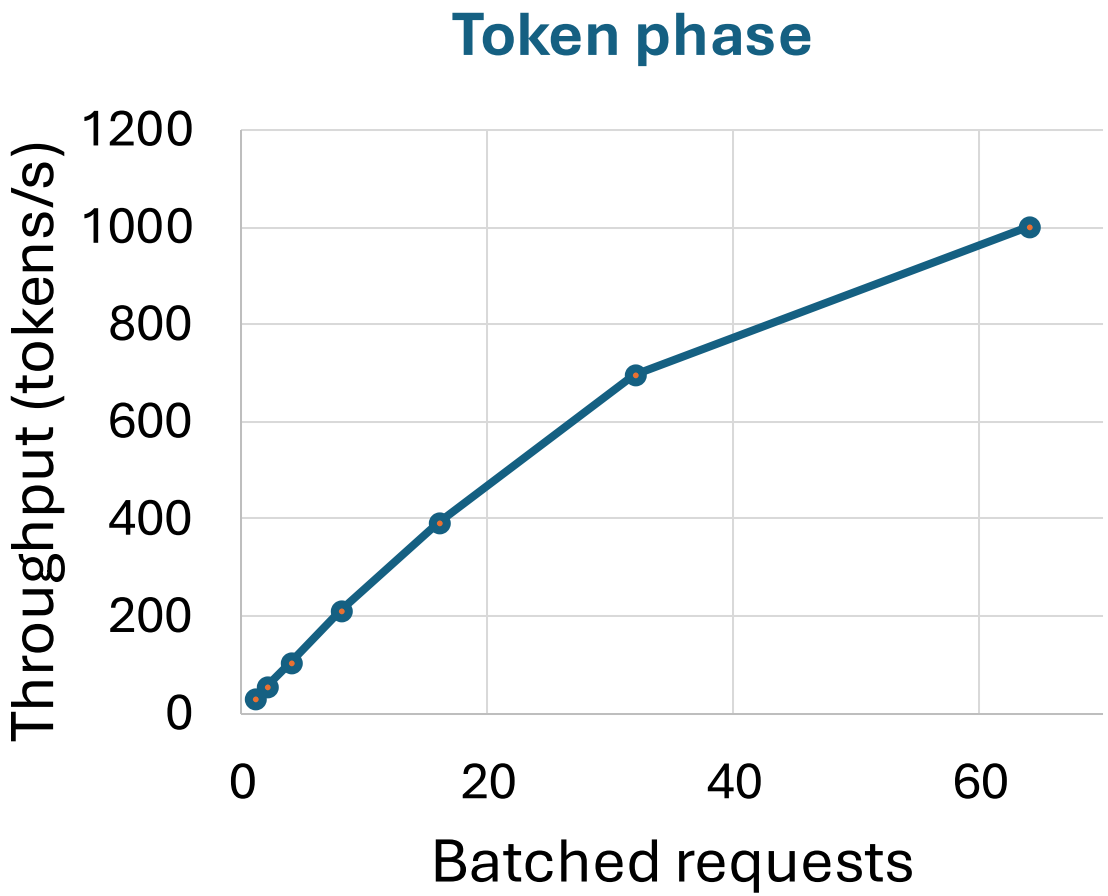
Prompt computation vs. token generation



Prompt phase	Token phase
User input processed in parallel	Serialized token generation
Compute intensive	Memory intensive (relies on KV cache)

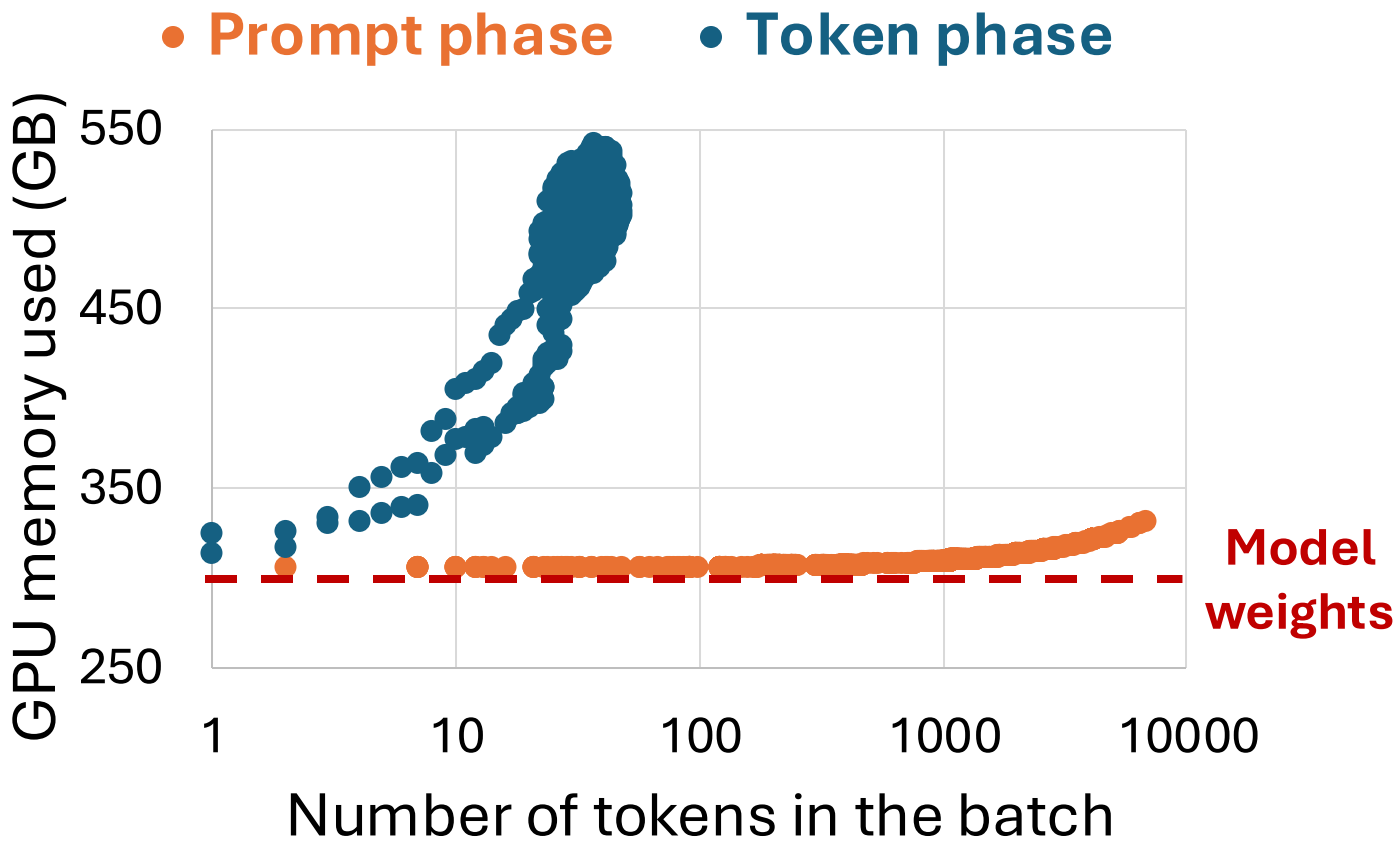
Prompt phases hit a throughput bottleneck





Profiled for BLOOM-176B on vLLM with 512 input size per request

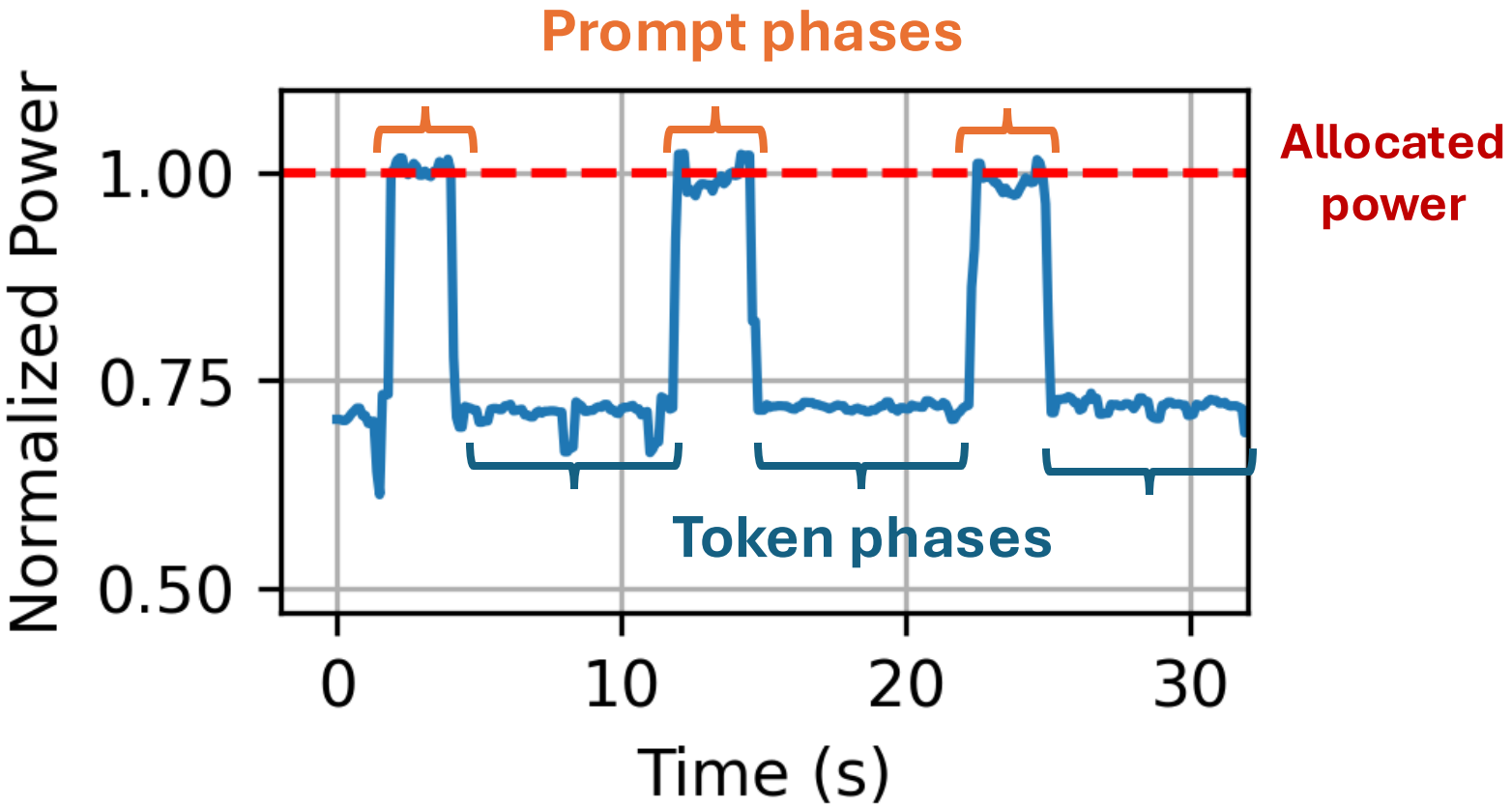
Token phase batches are memory constrained



Token phases use the KV cache, which can take up hundreds of GBs!

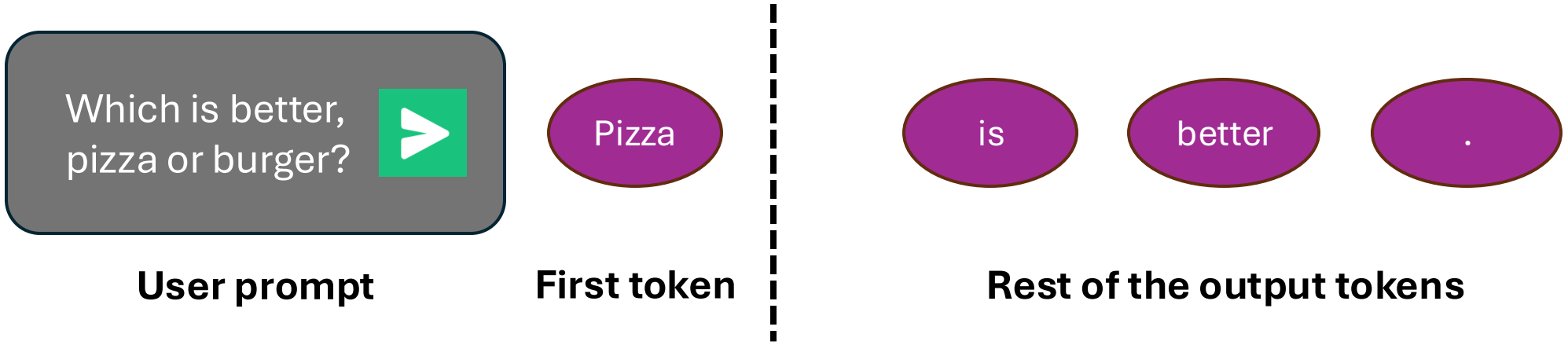
Profiled for BLOOM-176B on 8 A100 GPUs

Prompt phases are power intensive



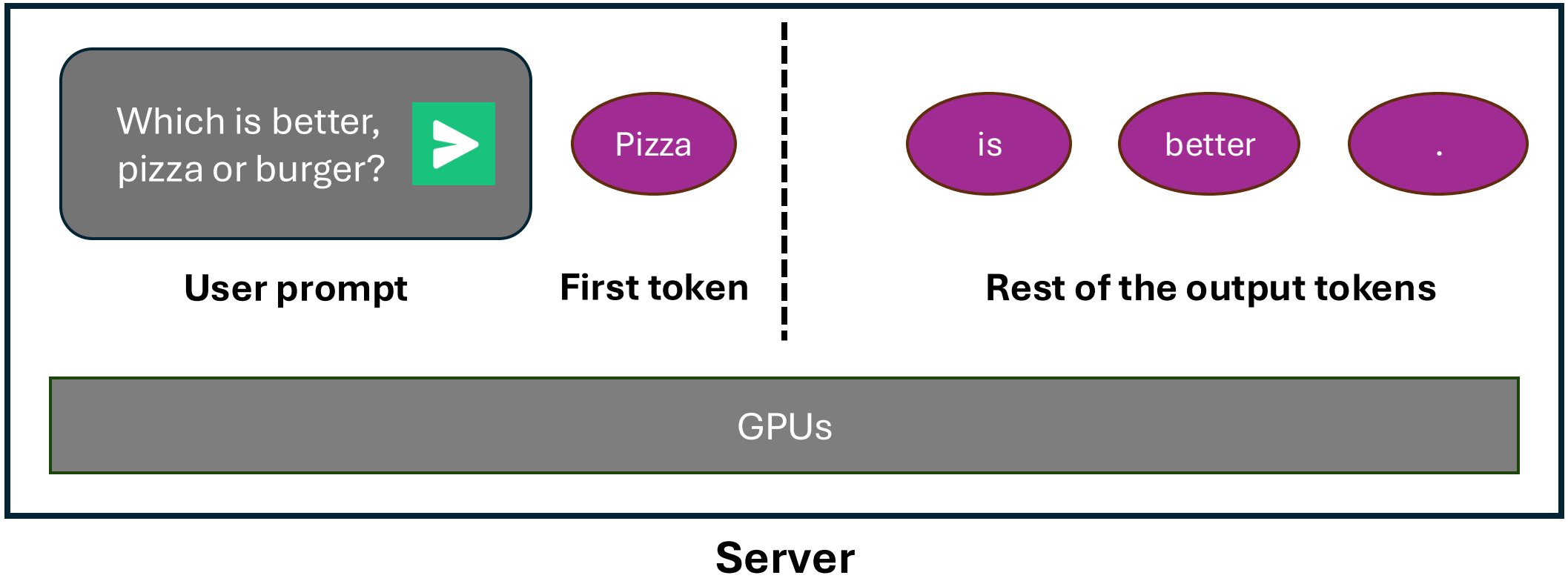
3x BLOOM-176B inference requests on 8 GPUs

Prompt computation vs. token generation

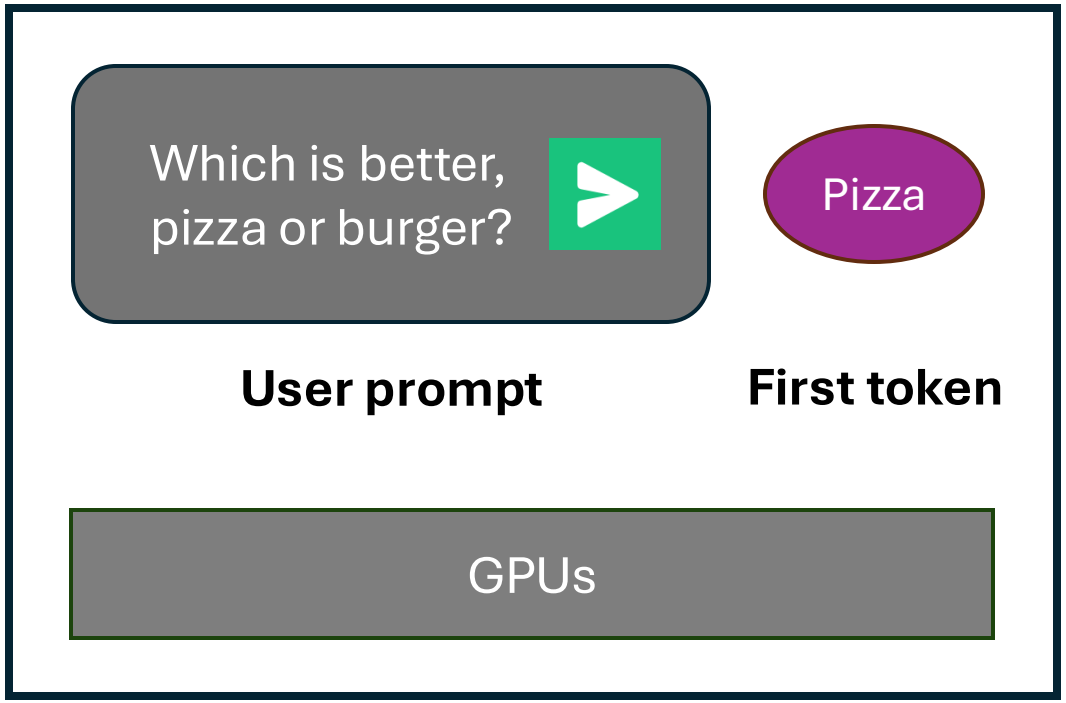


Prompt phase	Token phase
Compute and power intensive	Memory intensive
Limited batching benefits	Batching improves throughput

Inefficient to run both on the same hardware

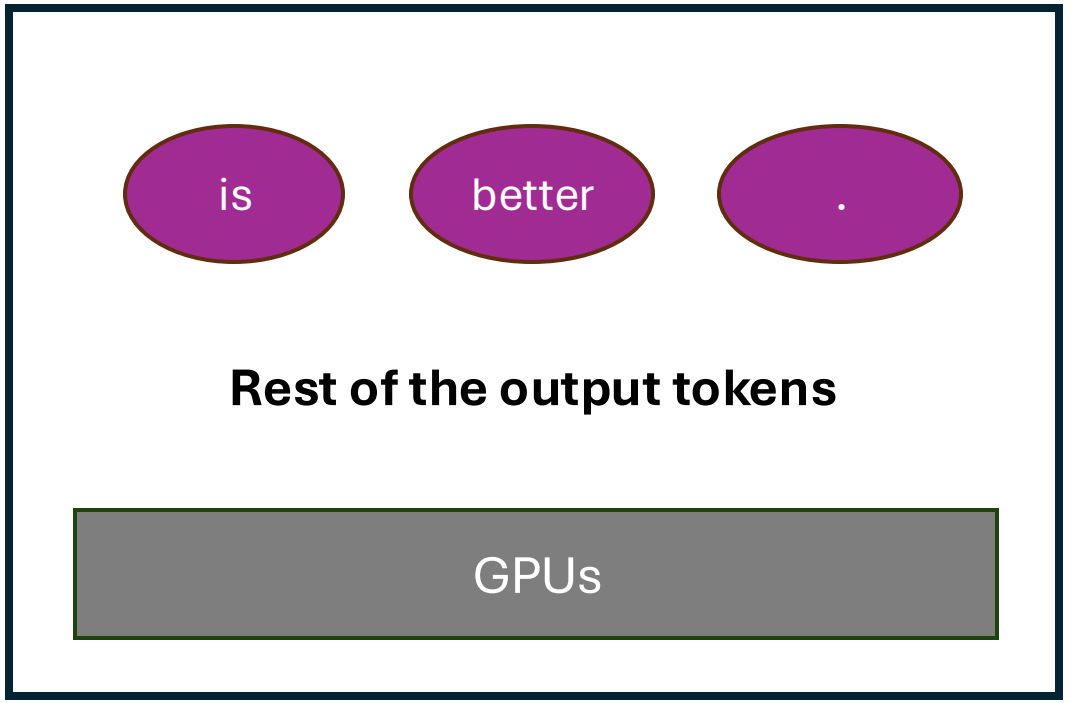


Splitwise splits phases onto different servers



Prompt server

Small batches, maximum power



Token server

Large batches, power capped

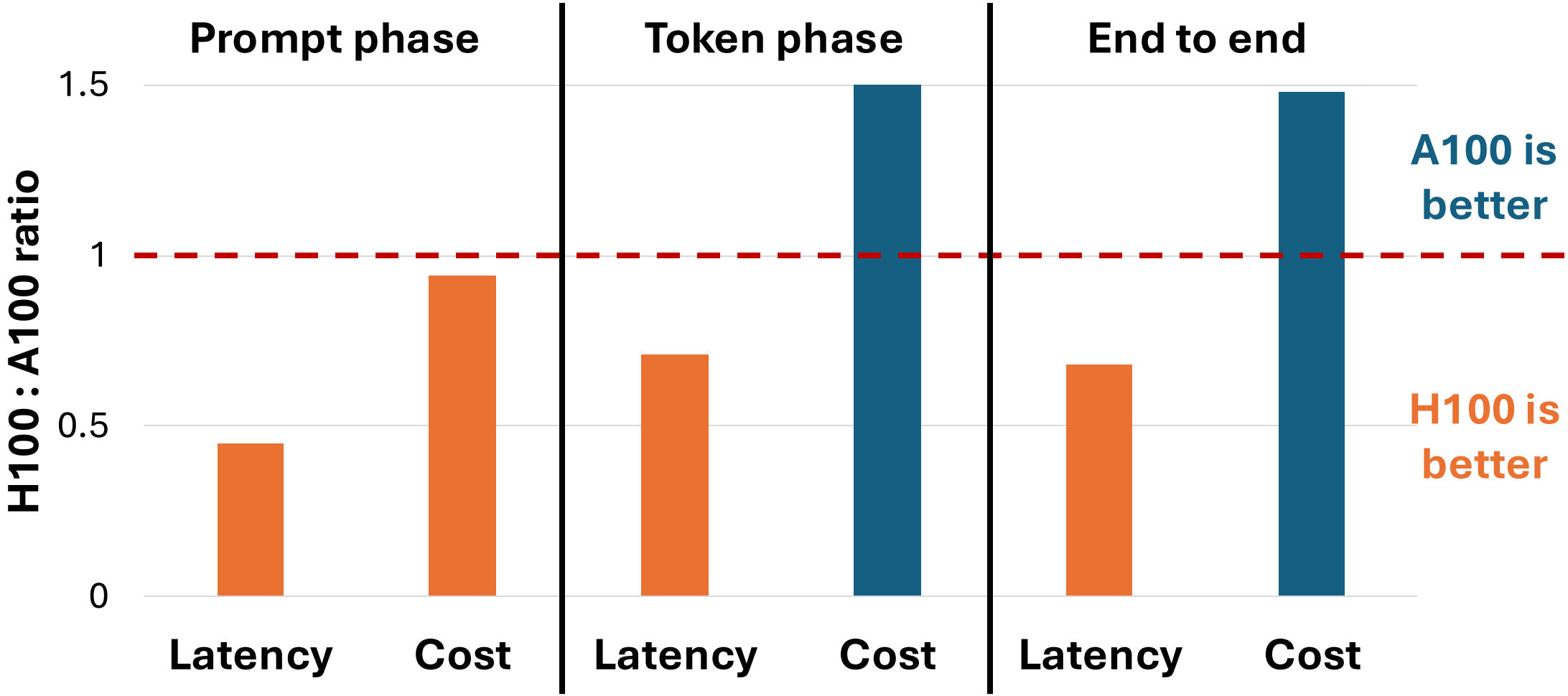
Different trade-offs on different GPUs

Spec	H100 : A100 ratio
Cost	2.15x
Max. power	1.75x
TFLOPs	3.43x

GPU memory scales slower than compute

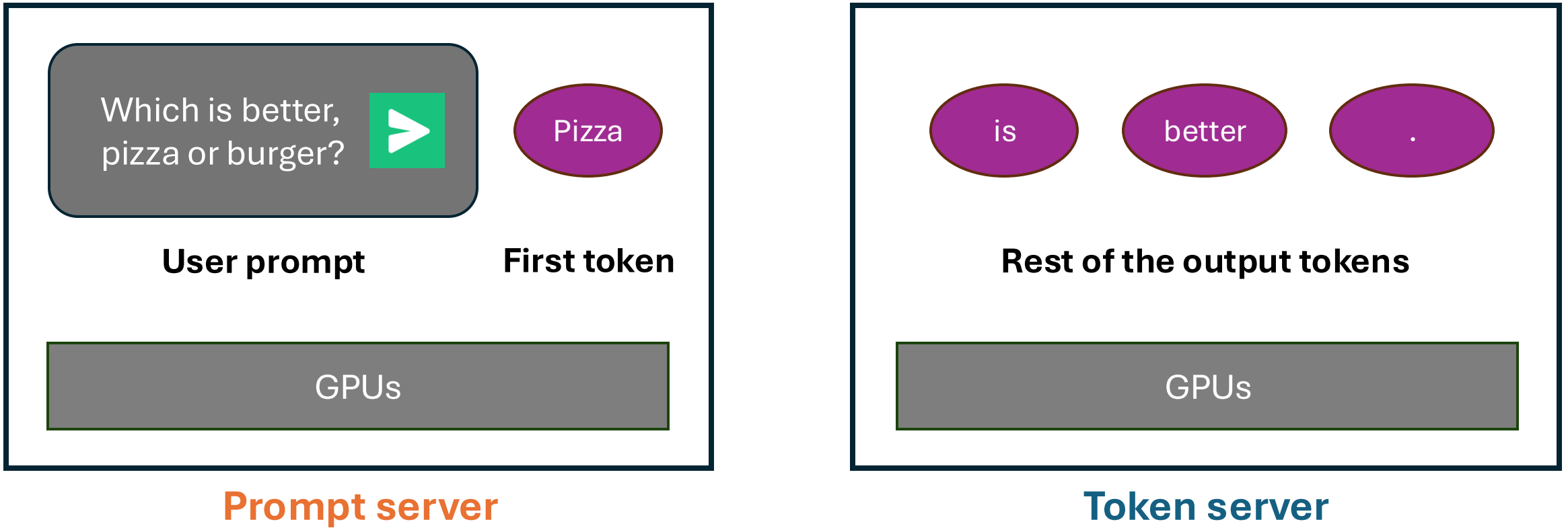
Spec	H100 : A100 ratio
Cost	2.15x
Max. power	1.75x
TFLOPs	3.43x
HBM capacity	1.00x
HBM bandwidth	1.64x

Phase preference for different GPUs

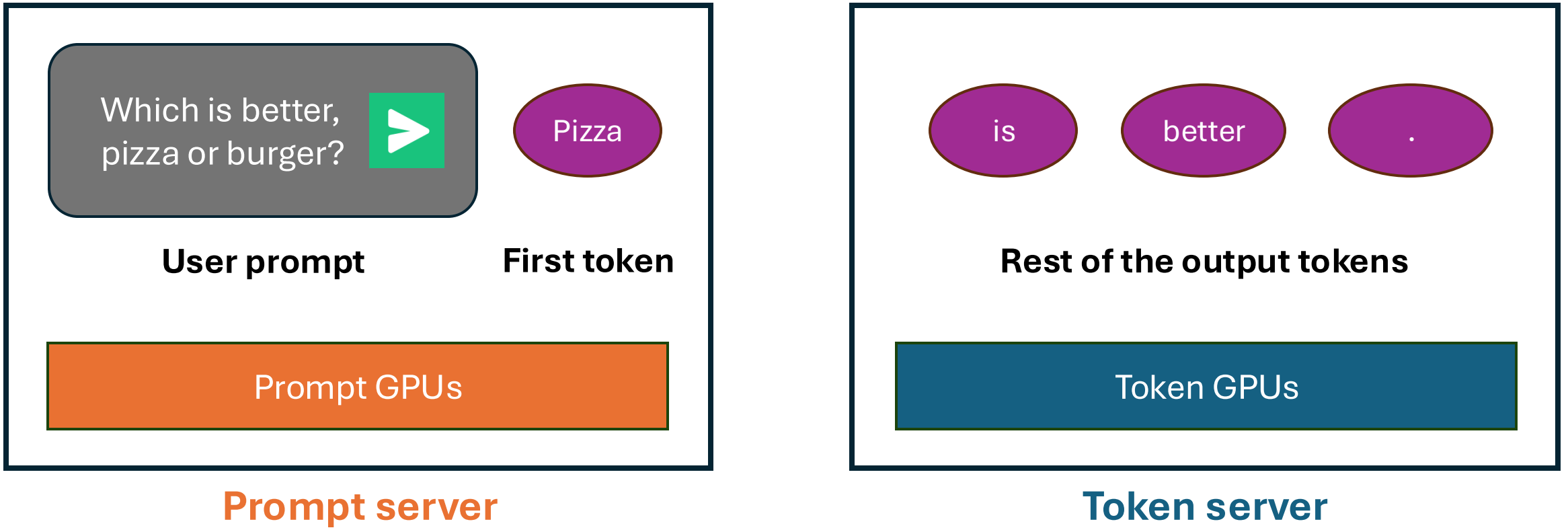


For BLOOM-176B on the mean request size in production

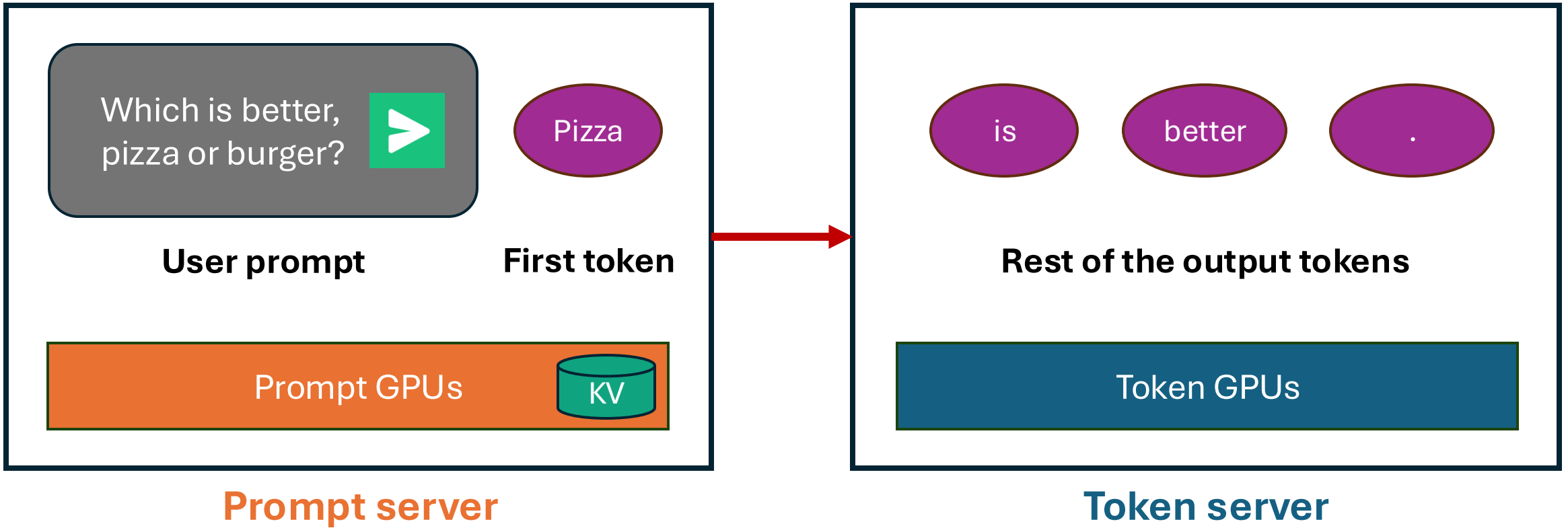
Splitwise splits phases onto different servers



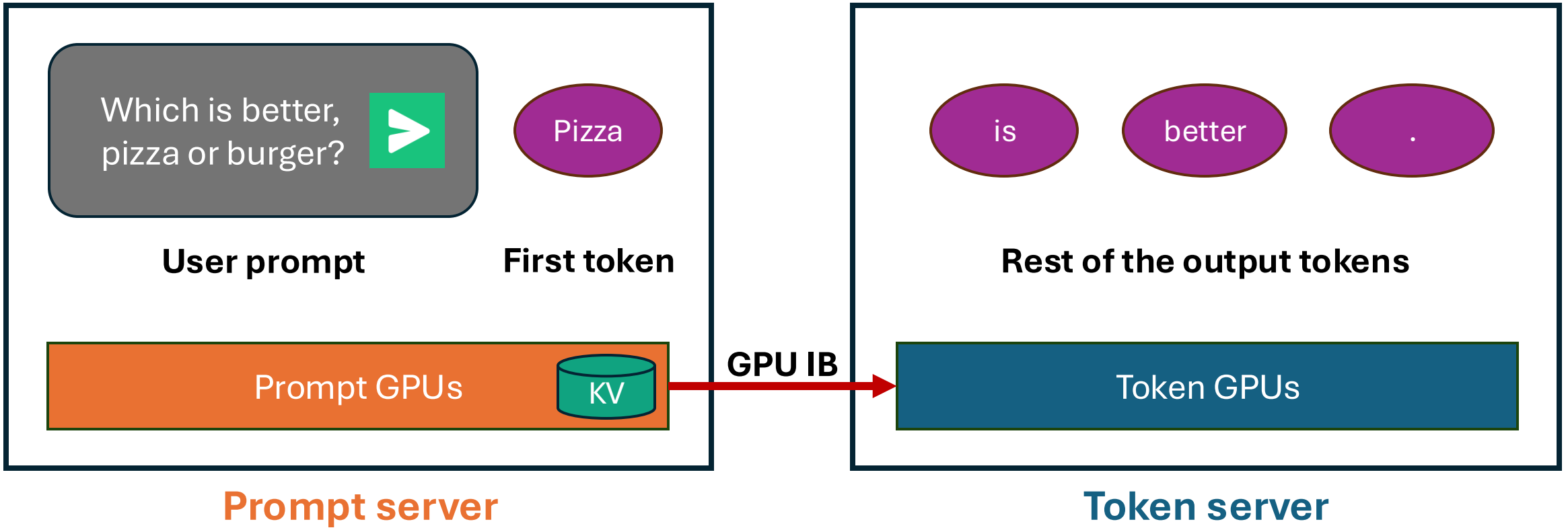
Each phase could use preferred hardware



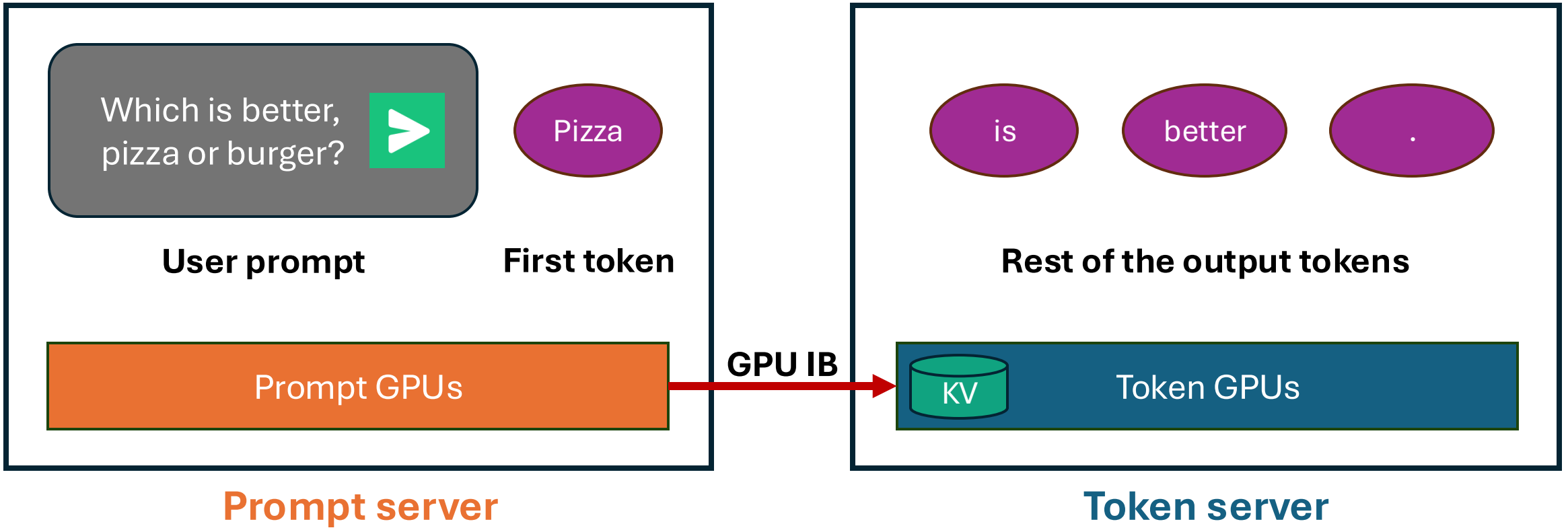
Splitting inference requires fast state transfers



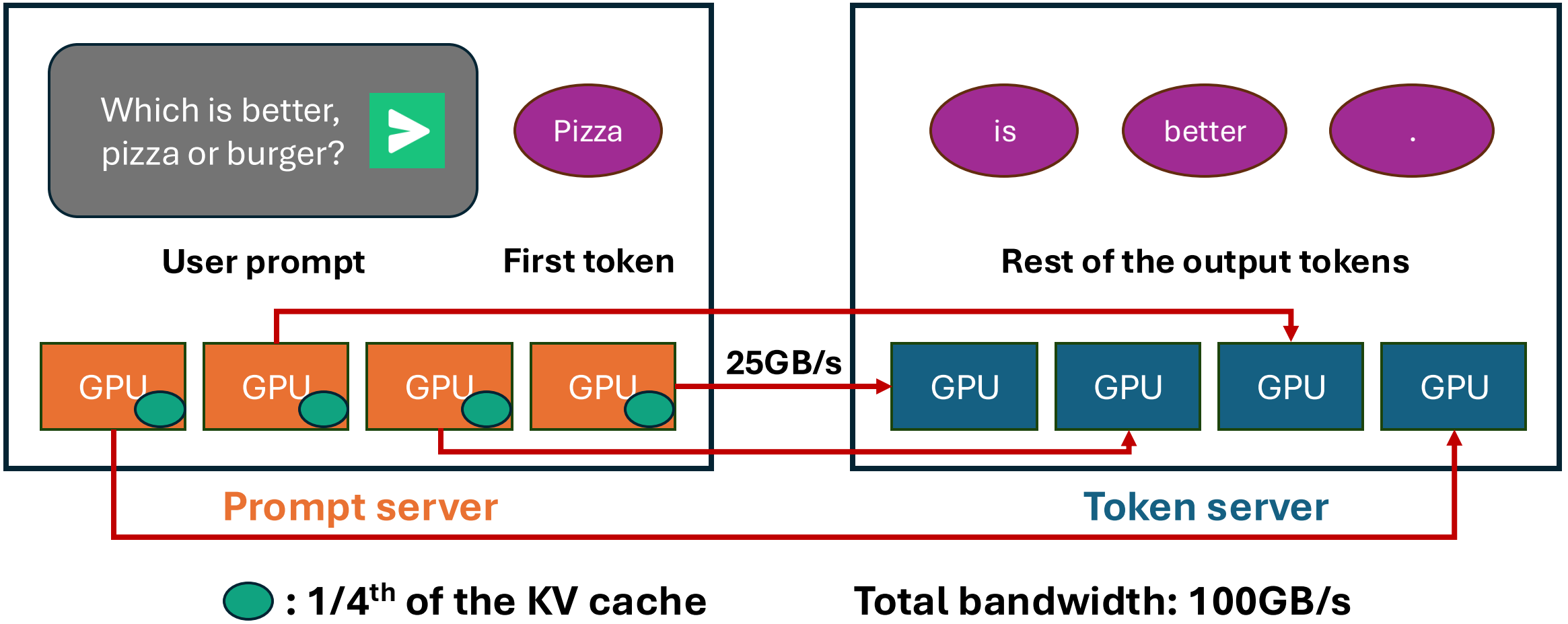
Splitwise uses GPU Infiniband to ship state



Splitwise uses GPU Infiniband to ship state

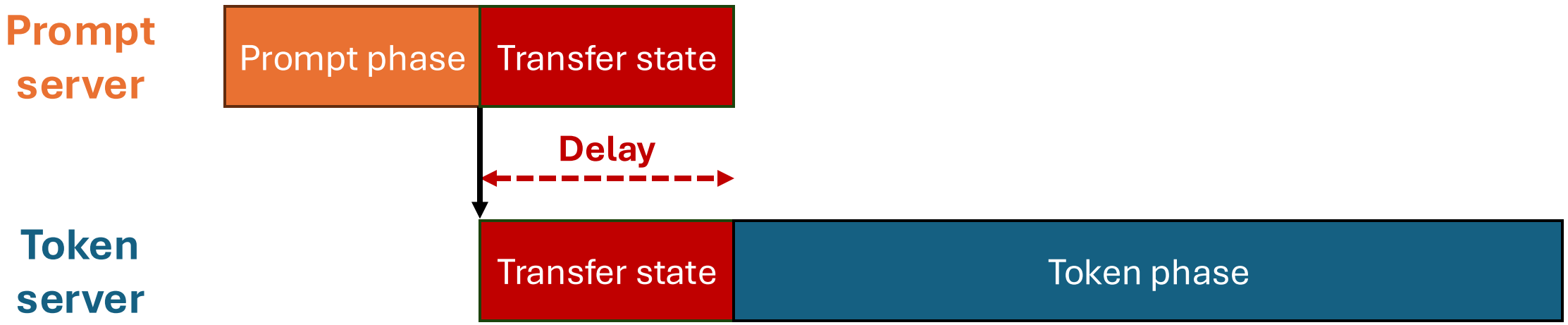


Parallelize transfers for high bandwidth



Assuming 4-way tensor parallelism

Transfer overheads may still be large

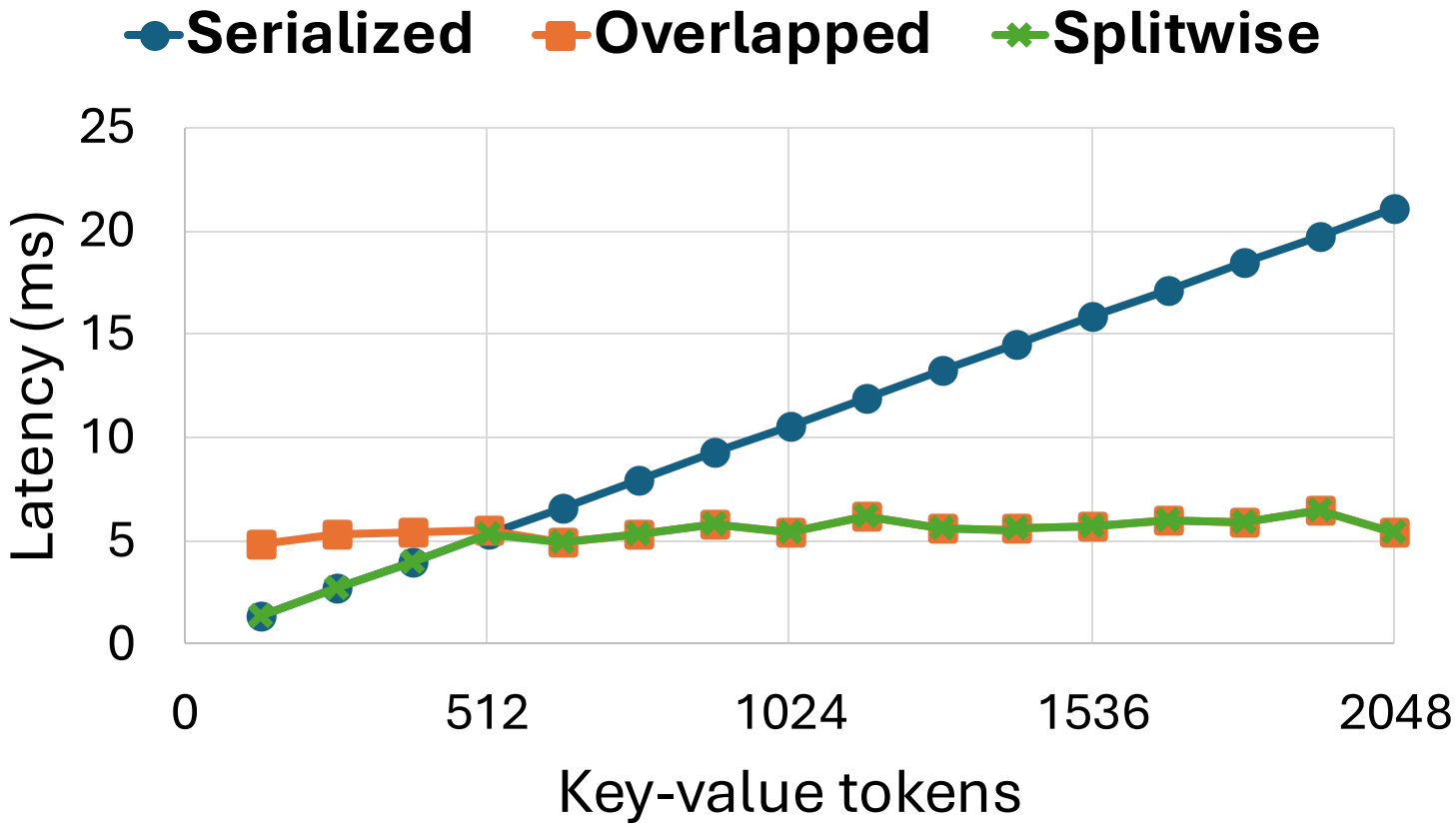


KV cache sizes can be hundreds of GBs!

Splitwise overlaps transfer with prompt phase

Start shipping the KV-cache after the first prompt layer

Splitwise adds very little latency overhead



Less than ~0.8% overhead for a typical inference request

Implemented in vLLM

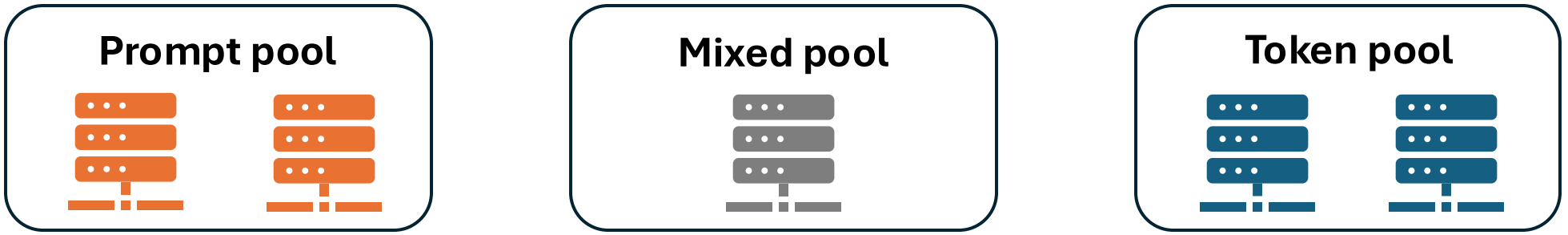
Splitwise: Phase Splitting for Generative LLMs

Characterize generative LLM inference

Split inference onto different servers

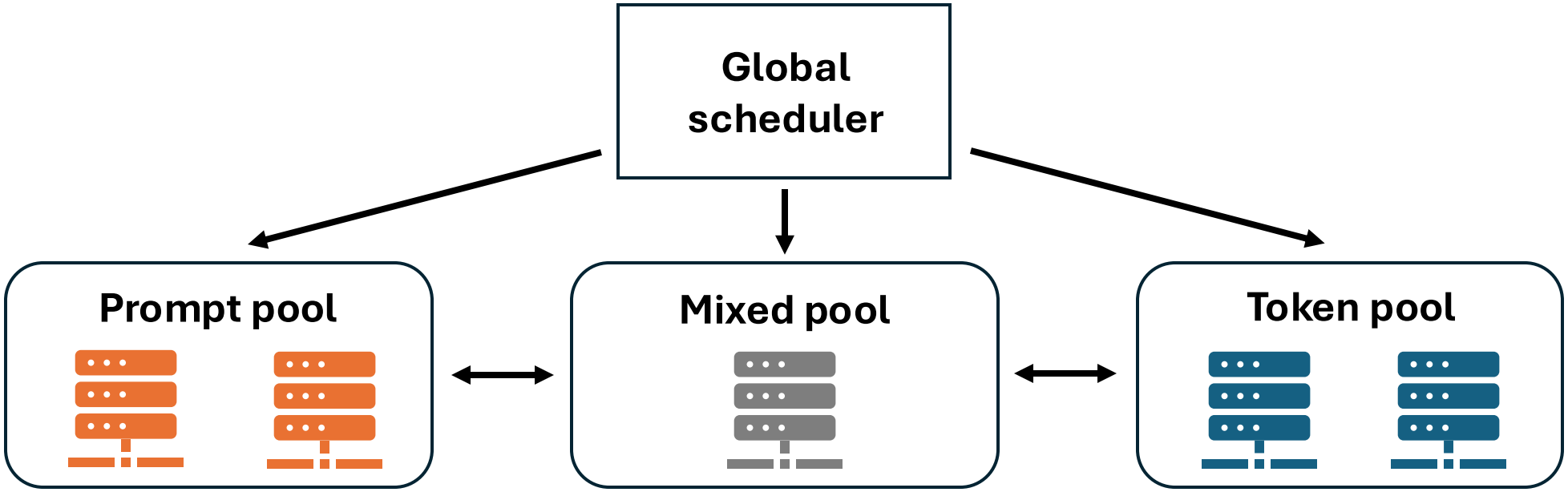
Design clusters using Splitwise

Splitwise partitions servers into three pools



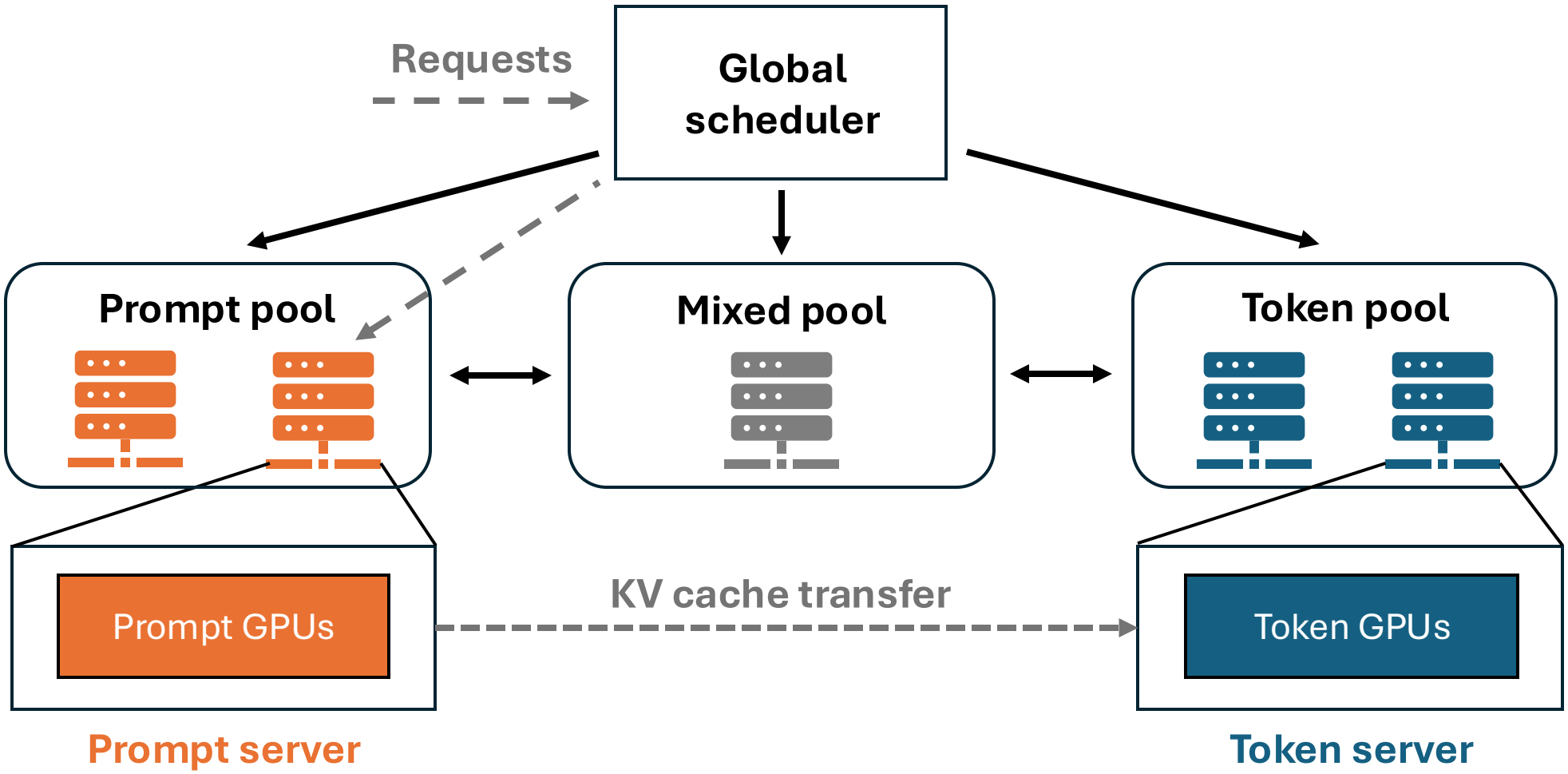
Details in the paper

Servers are fungible across the pools



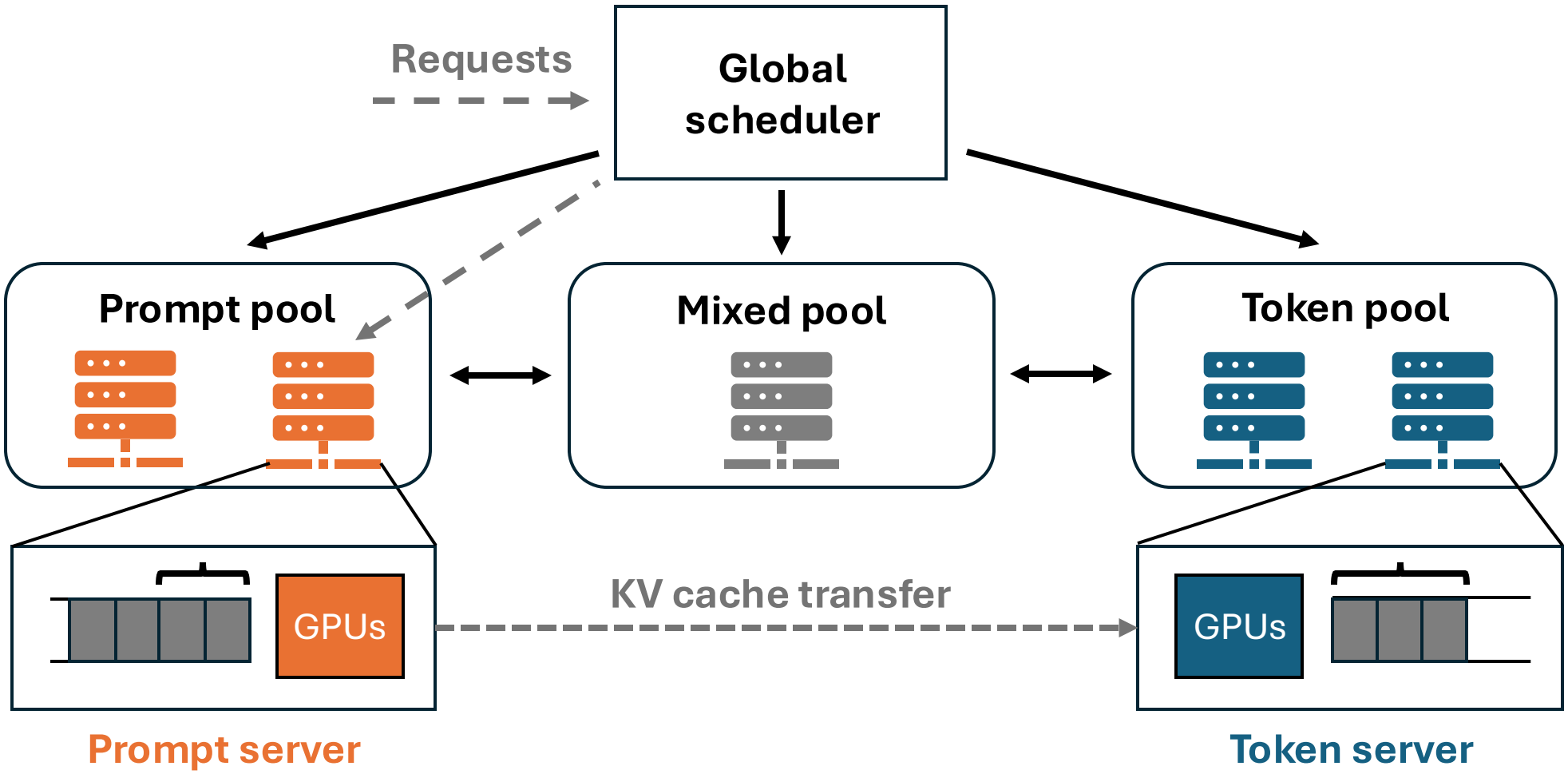
Details in the paper

Scheduler decides how to split LLM requests



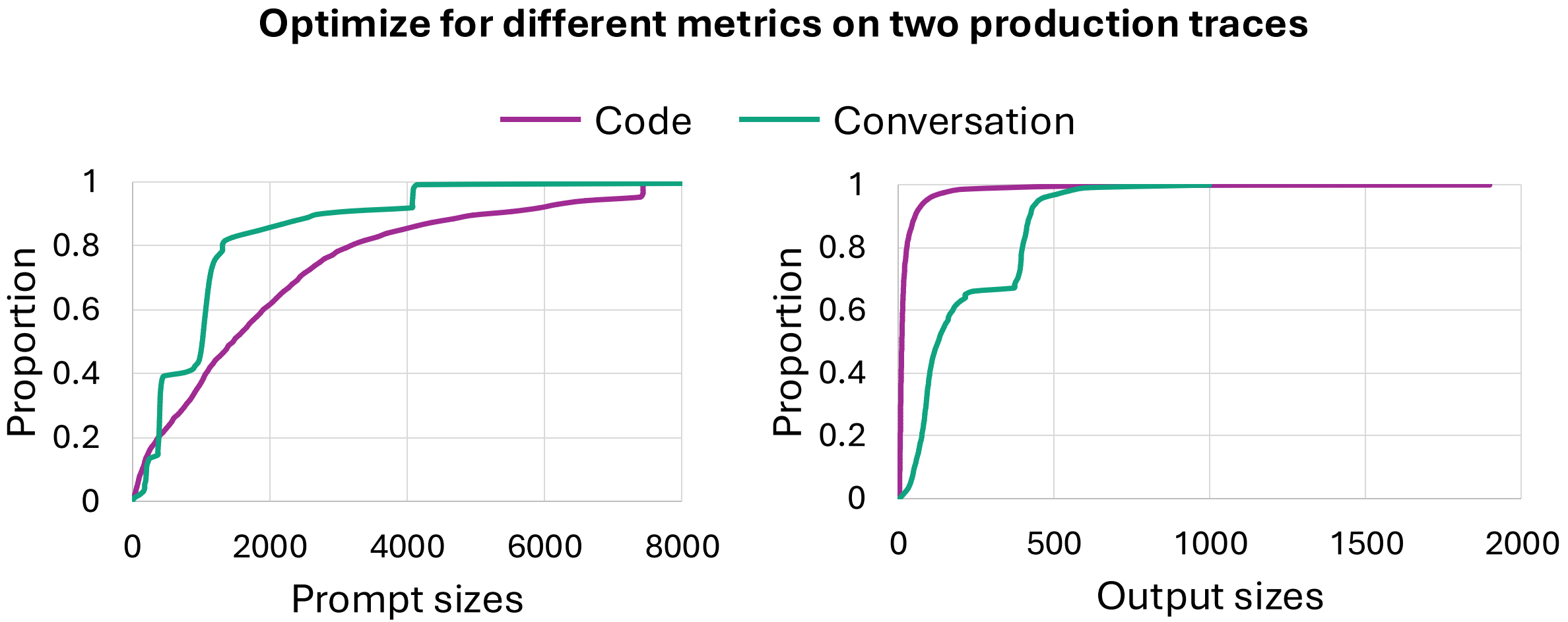
Details in the paper

Servers implement phase-aware batching



Details in the paper

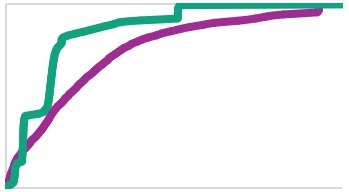
Evaluation compares different cluster designs



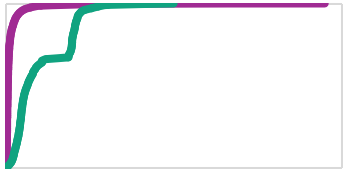
Traces available at: https://github.com/Azure/AzurePublicDataset

Simulated at scale using performance profiles

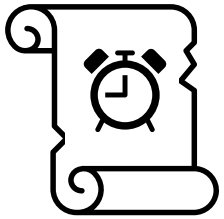
App. inputs



Prompt sizes

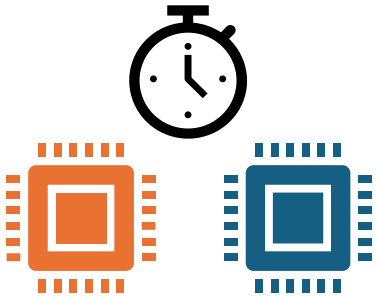


Output sizes

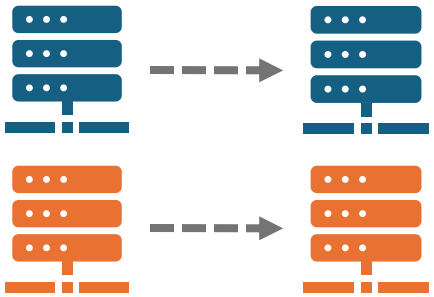


Service-Level Objectives

Hardware profiles

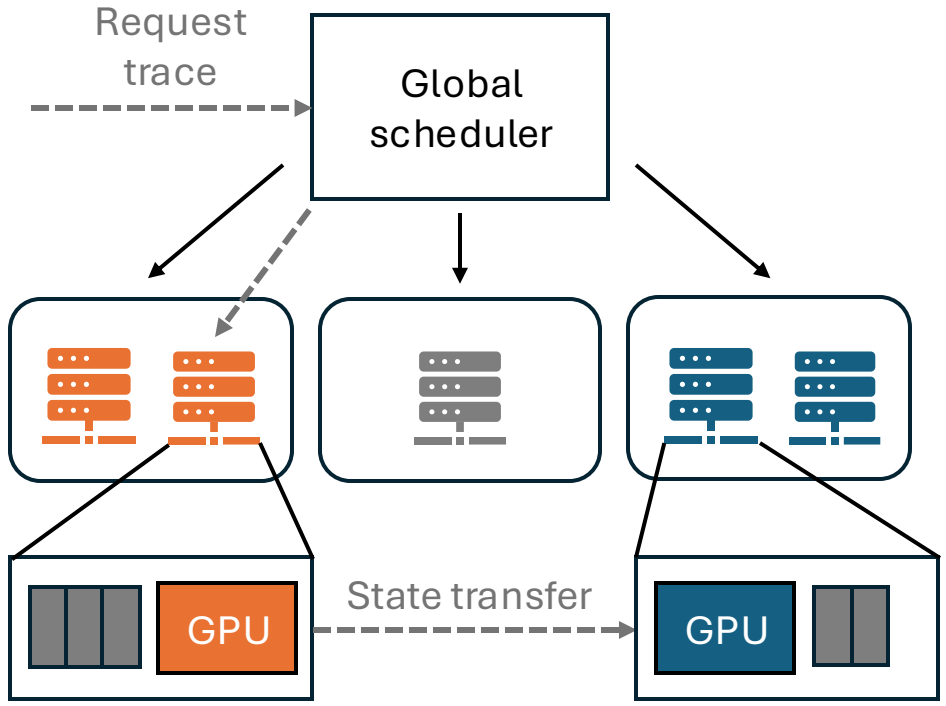


LLM performance



KV-cache transfer

Simulated cluster

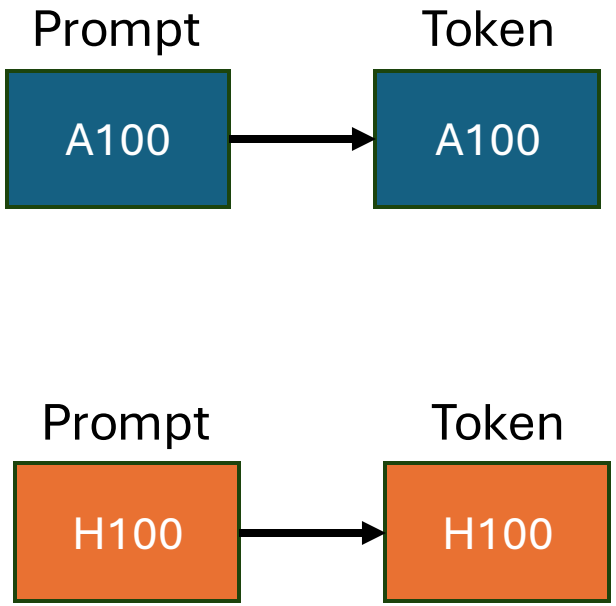


Baselines

Run requests end-to-end on same server

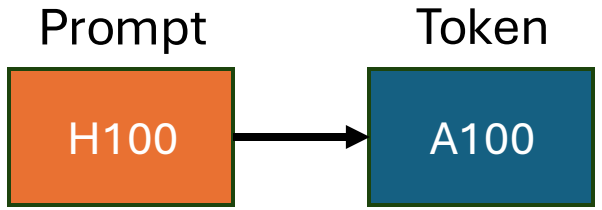
Splitwise
homogeneous

Use the same server type for prompt and token phases



Splitwise
heterogeneous

Use H100s for prompt and A100s for token phases

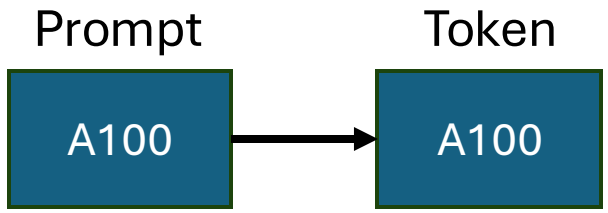


Baseline

Run requests end-to-end on same server

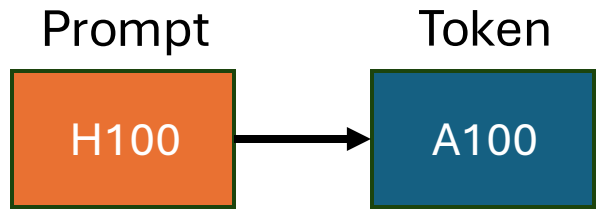
Splitwise
homogeneous

Use the same server type for prompt and token phases

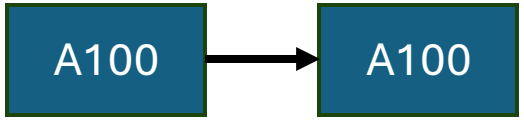
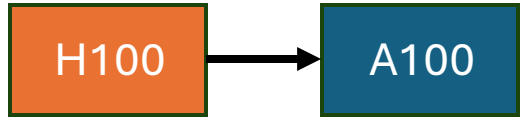


Splitwise
heterogeneous

Use H100s for prompt and A100s for token phases



More results in the paper

Throughput optimized clusters	Baseline	Splitwise homogeneous	Splitwise heterogeneous
	 x70	 x45 x25	 x25 x26
#Servers	1x	1x	0.73x
Cost	1x	1x	1.14x
Power	1x	1x	1x
Throughput	1x	2.4x	2.6x

Cost optimized clusters	Baseline	Splitwise homogeneous	Splitwise heterogeneous
	A100 x88	A100 x25 → A100 x16	H100 x11 → A100 x19
#Servers	1x	0.46x	0.34x
Cost	1x	0.46x	0.48x
Power	1x	0.46x	0.43x
Throughput	1x	1x	1x

Splitwise

Phase Splitting for Efficient Generative LLM Inference

LLM inference requests have distinct prompt and token phases

Splitting inference enables phase-specific resource management

Splitwise improves inference cluster efficiency across various metrics

Paper, code, traces at

aka.ms/splitwise

Thanks!

pratyush@cs.uw.edu