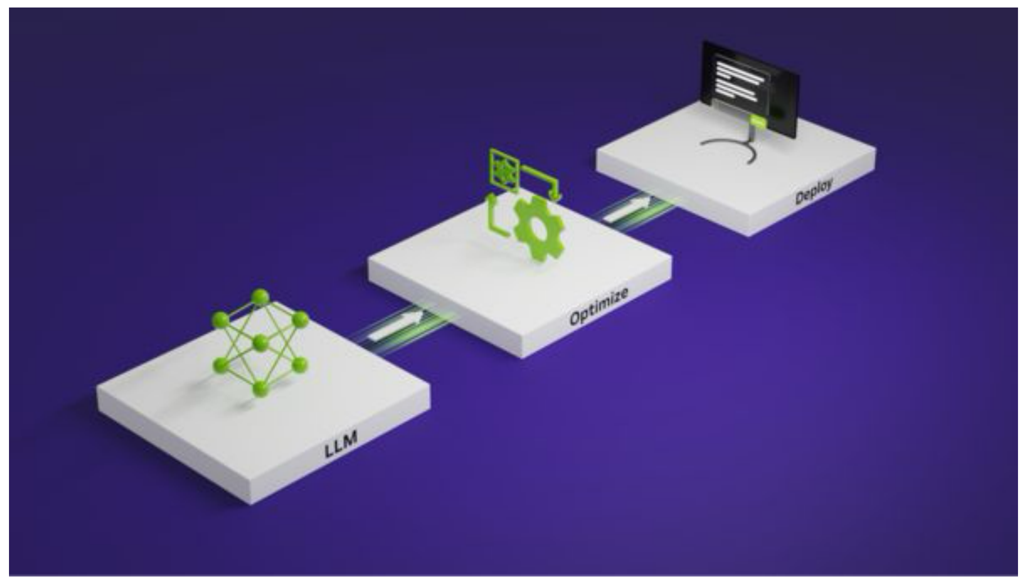
Taming Throughput-Latency Tradeoff in LLM Inference With Sarathi-Serve

Authors: Amey Agarwal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav Gulvani, Alexey Tumanov, Ramachandran Ramjee

Presenter: Sakthi Karimanal



Rise of LLMs

Technology

ChatGPT sets record for fastest-growing user base - analyst note

By Krystal Hu

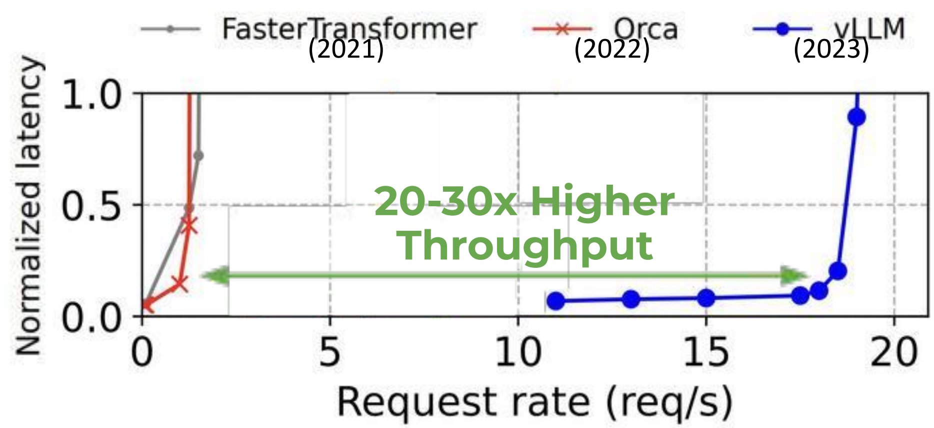
February 2, 2023 7:33 AM PST · Updated a year ago

Aa

CLIMATE

Google's carbon emissions surge nearly 50% due to AI energy demand

PUBLISHED TUE, JUL 2 2024·3:41 PM EDT | UPDATED MON, JUL 8 2024·9:32 AM EDT



& Inference Systems

Can we maintain low latency with high throughput?

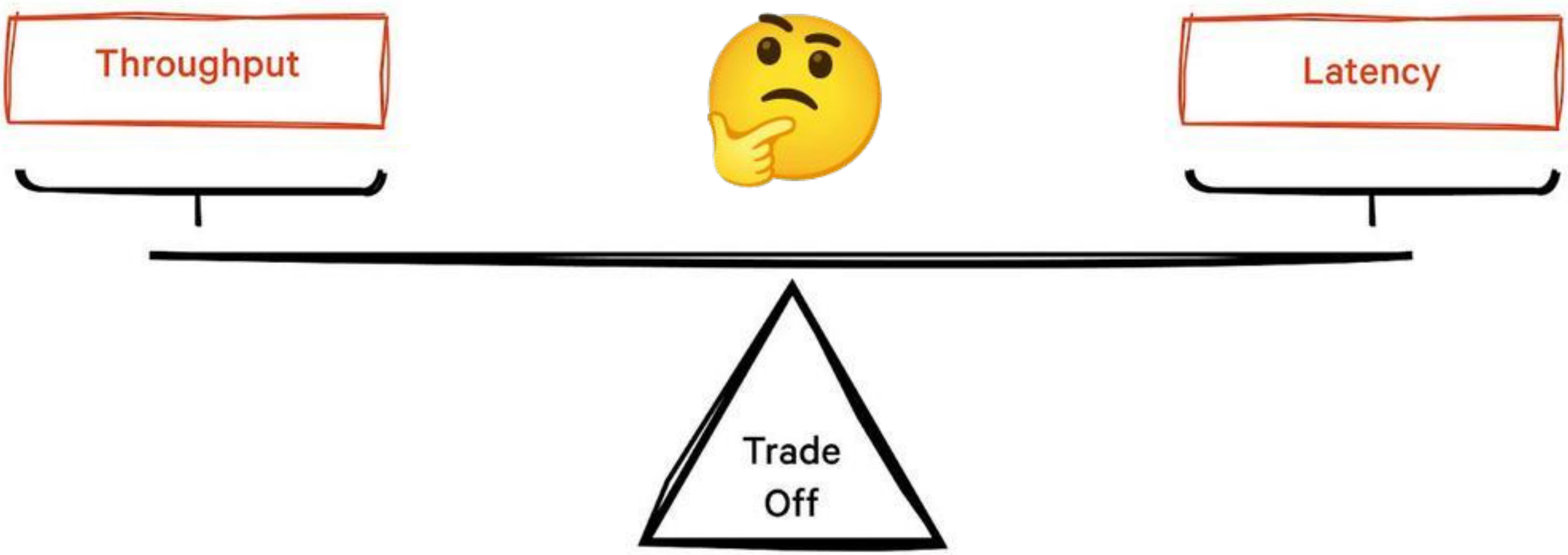


Image credits: Redpanda

Demo 📺

In this talk...

🔬 **Latency-throughput tradeoff:** Analyzing LLM batching policies

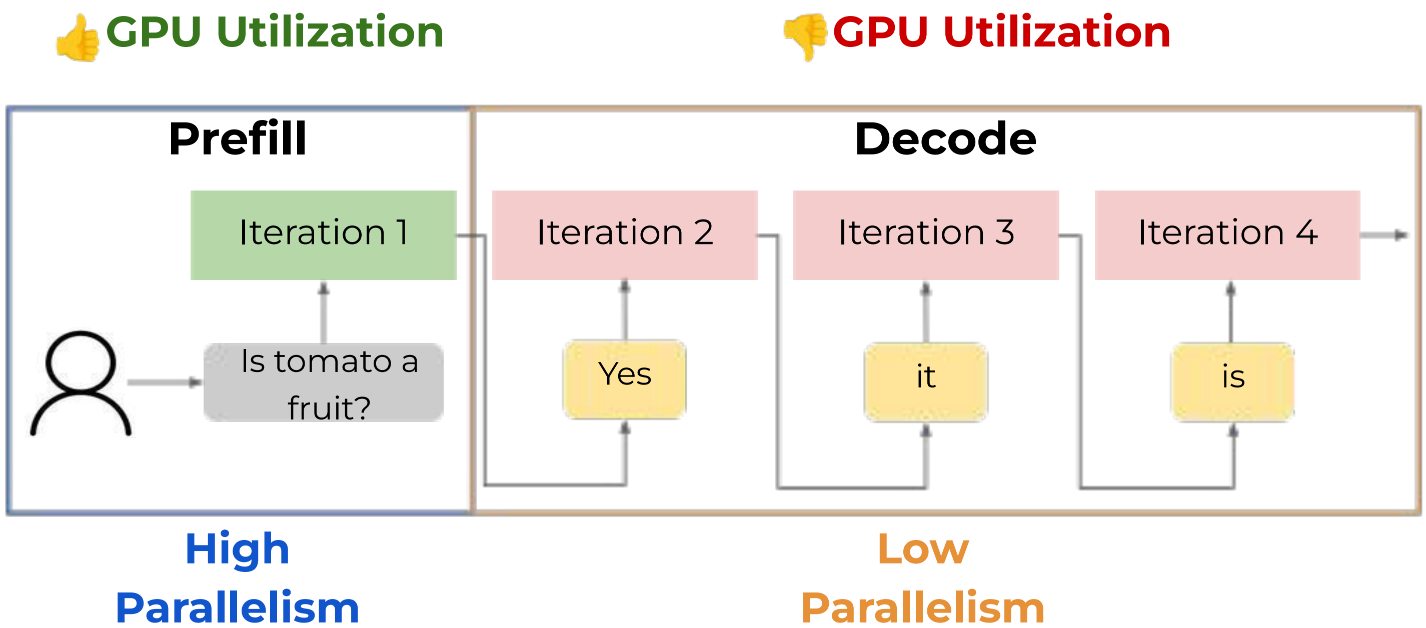
🥗 **Finding a free lunch:** Arithmetic Intensity Slack in LLM Inference

🧠 **Stall-free batching:** Leveraging chunked prefill to overcome the latency-throughput tradeoff

🧮 **Evaluations:** Key results and analysis

What causes the latency-throughput tradeoff in LLM inference systems?

🔄 Background: LLM Inference 101



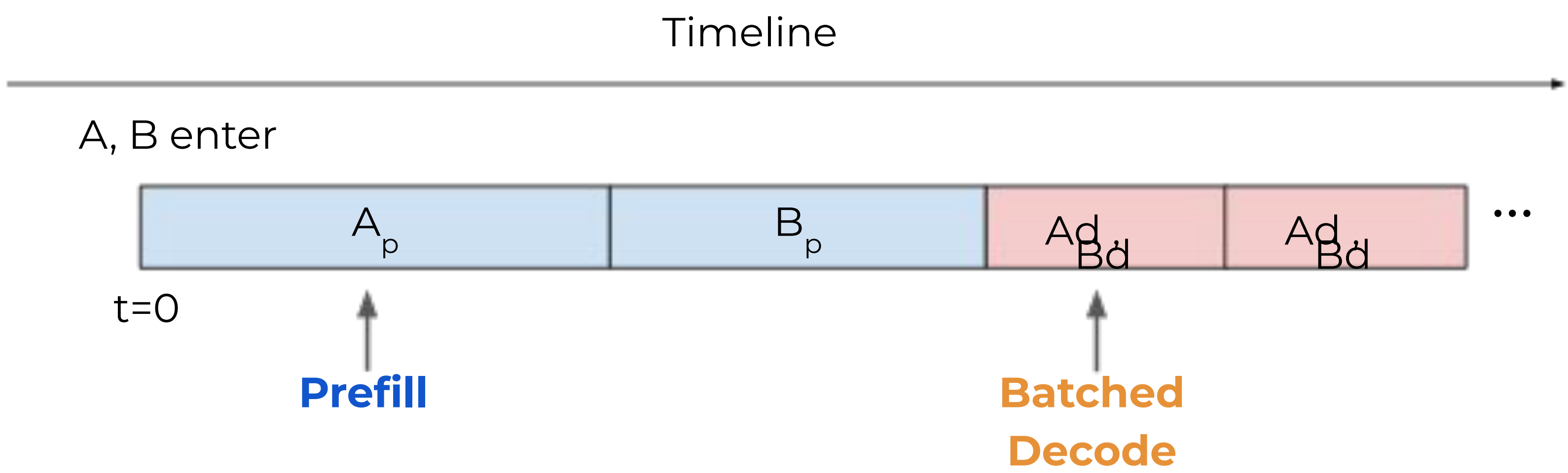
Background: LLM Inference Process

- Token passed from prefill phase through decoder to generate the next token until EOS
- Decode phase requires a full forward pass of the model
  - Leads to low compute utilization
- Multi-GPU Inference
  - Tensor-Parallelism splits the model weights and KV-cache equally across GPU workers
  - Pipeline-Parallelism splits the model layer-wise where each GPU is responsible for a subset layers
- Scheduling Policies
  - Prefill and Decode Prioritizing
  - Iteration-Level batching

How to improve parallelism during decode phase? 🤔

Batching

🥞 Background: Batching LLM Inference



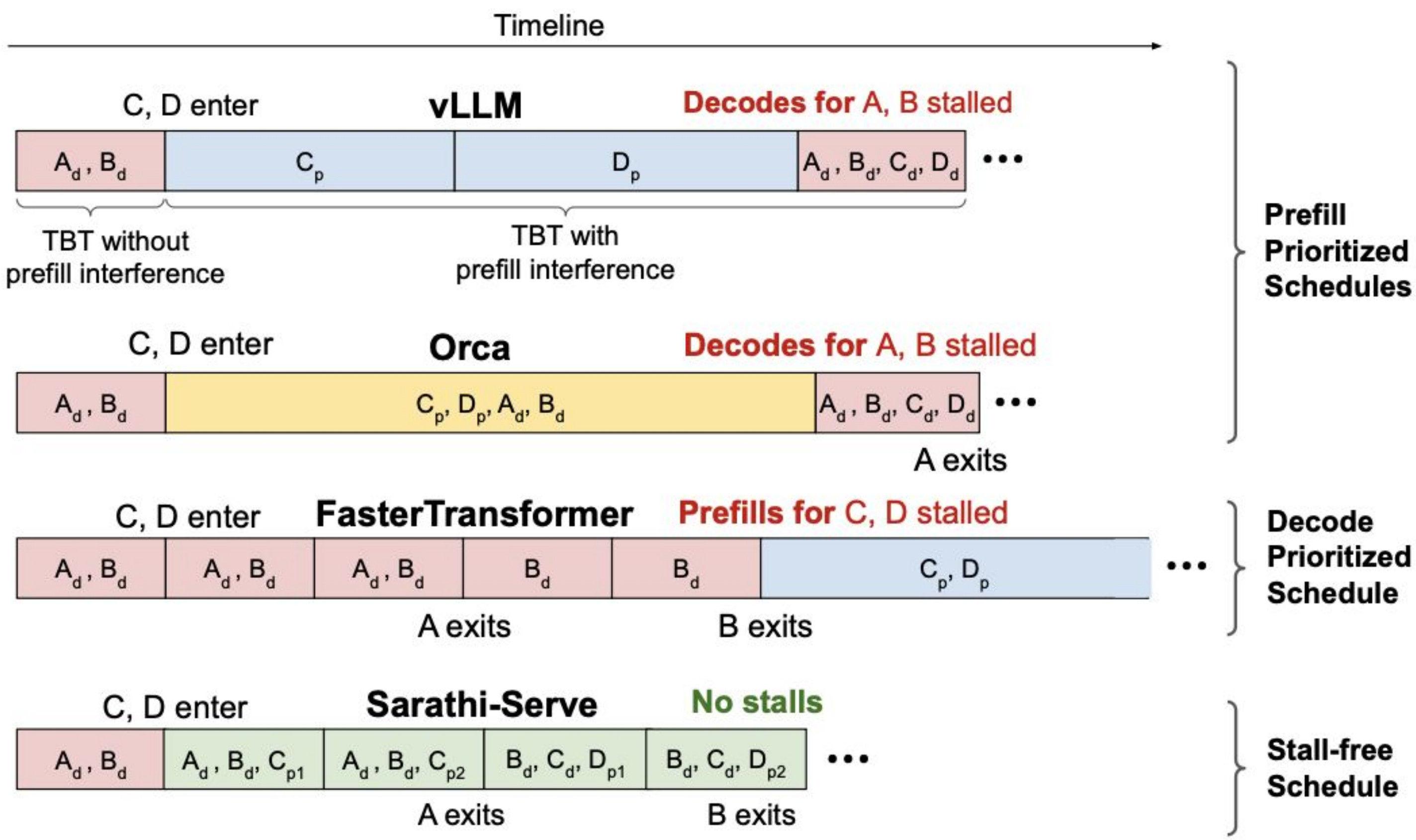
Decode efficiency increases linearly with batch size 🚀

🔼 **Batch size** ⇒ 🔼 **Throughput**

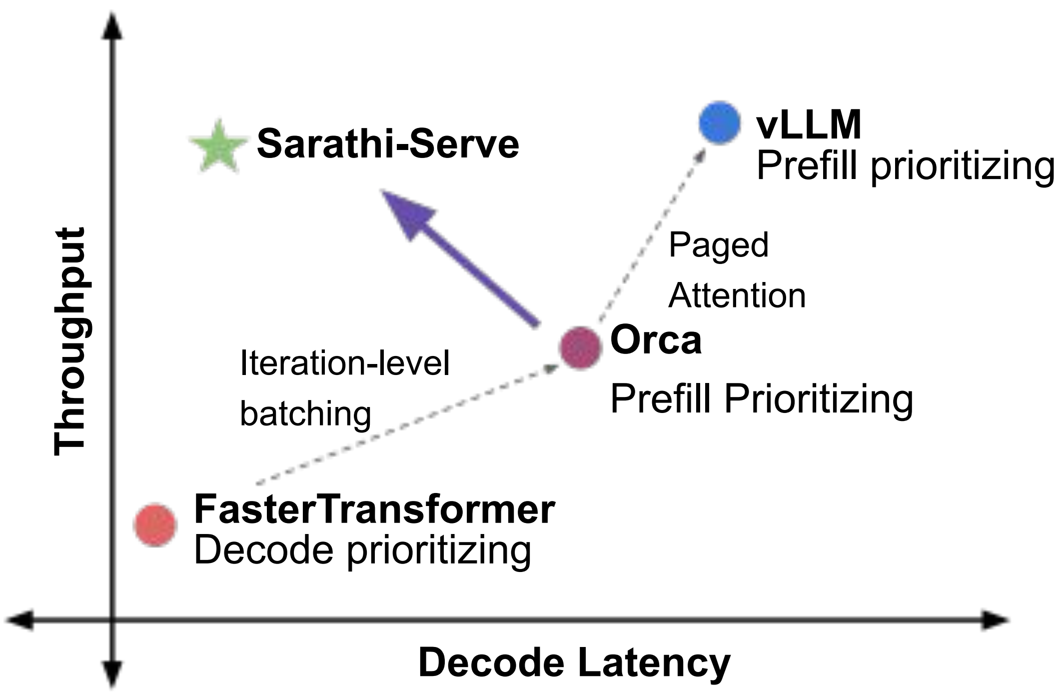
Motivation

- Low compute utilization during decodes
- Throughput-Latency Trade-off
  - Request-level batching has low TBT latency at cost of throughput
  - Iteration-level has higher throughput but more batches lead to latency spikes
- Pipeline bubbles waste GPU cycles
  - When waiting for micro batches to complete in prior stages
  - Can also arise due to non-uniform batch execution times

🔎 The Prefill-Decode Scheduling Conundrum



⚖️ The Latency-Throughput Tradeoff



Existing batching policies make a harsh latency-throughput tradeoff ❗

How can be we achieve both high throughput and low-latency? 🤔

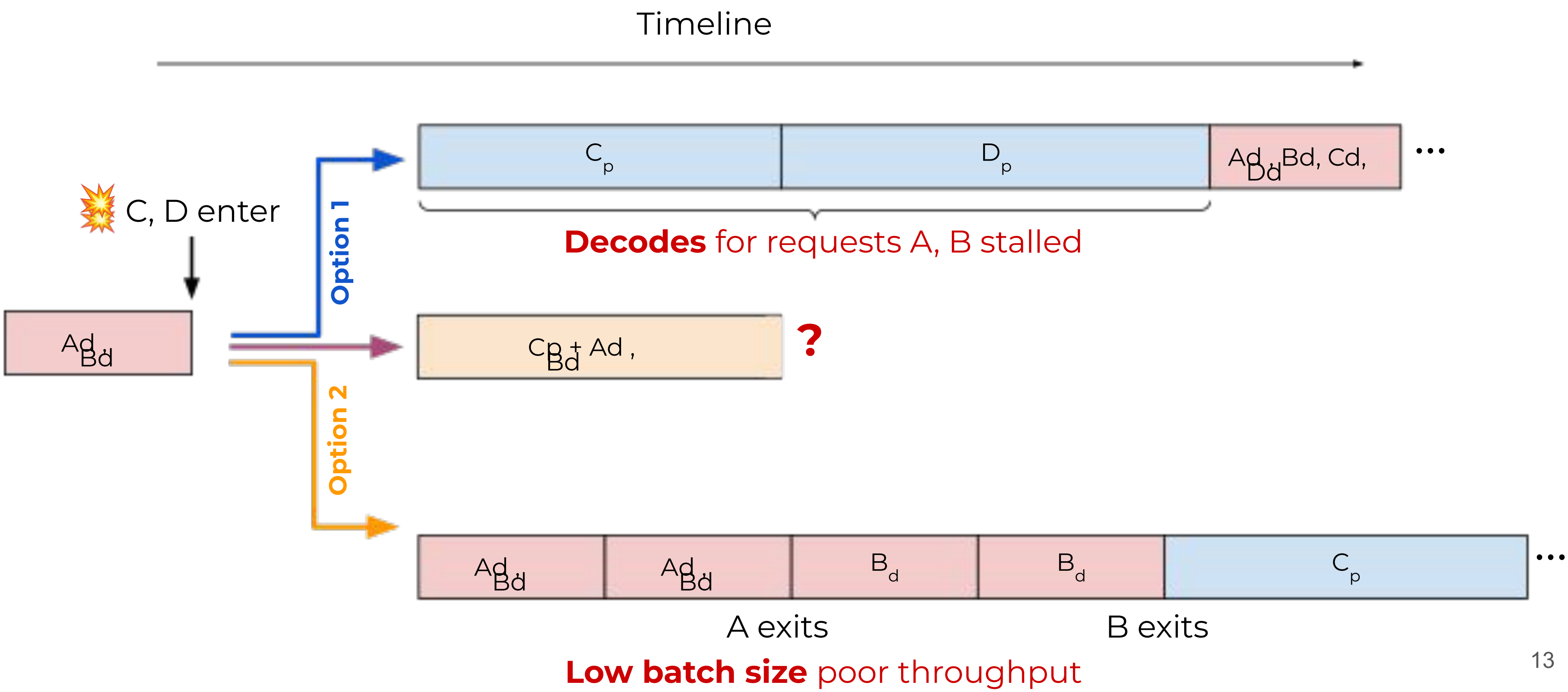
Sarathi-Serve

- Iteration-level scheduler
- Chunked-Prefills and Stall-free batching
    - Determine token budget
- SFB and chunked-prefills ensure uniform compute in hybrid batches
    - Reduces bubbles for PP, leading to scalable deployments
- Implemented on top of the open-source implementation of vLLM
- Used FlashAttention v2 and FlashInfer kernels for paged chunk prefill

Determining Token Budget

- TBT minimization
  - Smaller token budget is preferable
- Chunked-prefill computation
  - Overhead from memory reads
- Tile-quantization effect also affects the choice of token budget
  - Happens when matrix dimensions aren’t divisible by the tile size so thread blocks perform extraneous computation
- Hardware properties can affect token budget as well

The Prefill-Decode Scheduling Conundrum



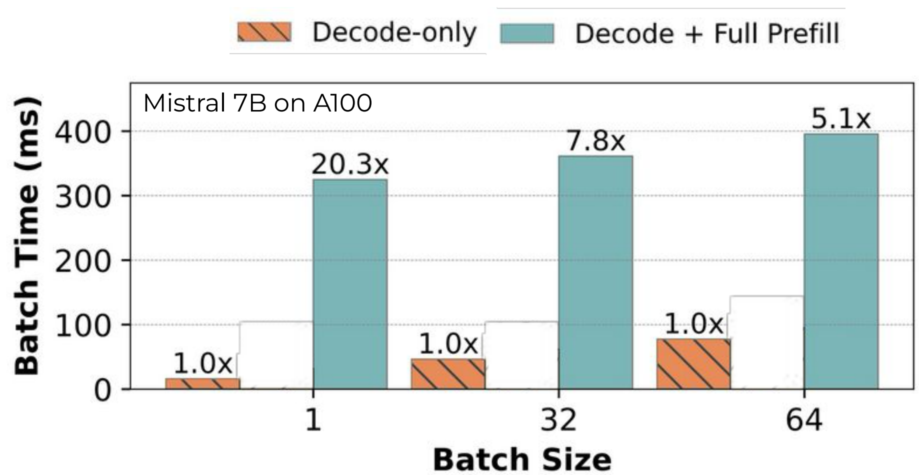
🔎 Mixed Batching

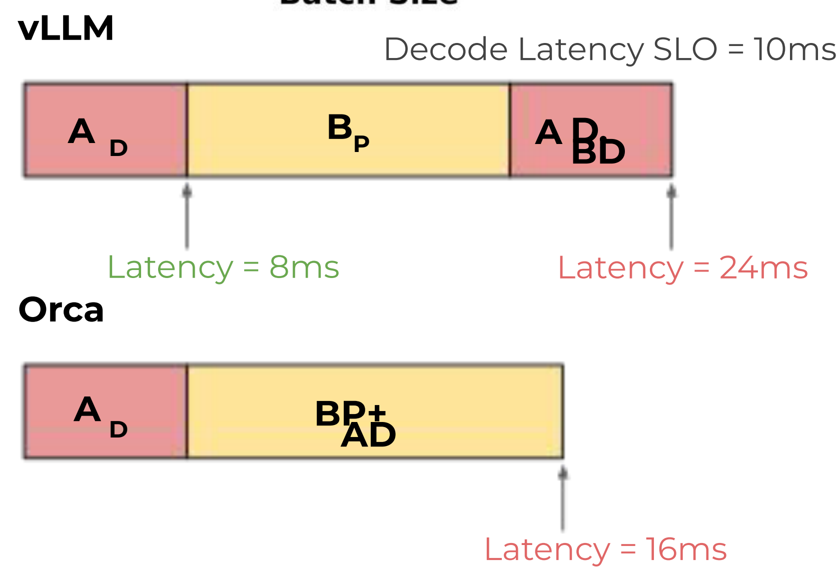
Idea

Fused computation of prefill and decodes

Challenge

Naively combining prefill and decode operations leads to increase in latency



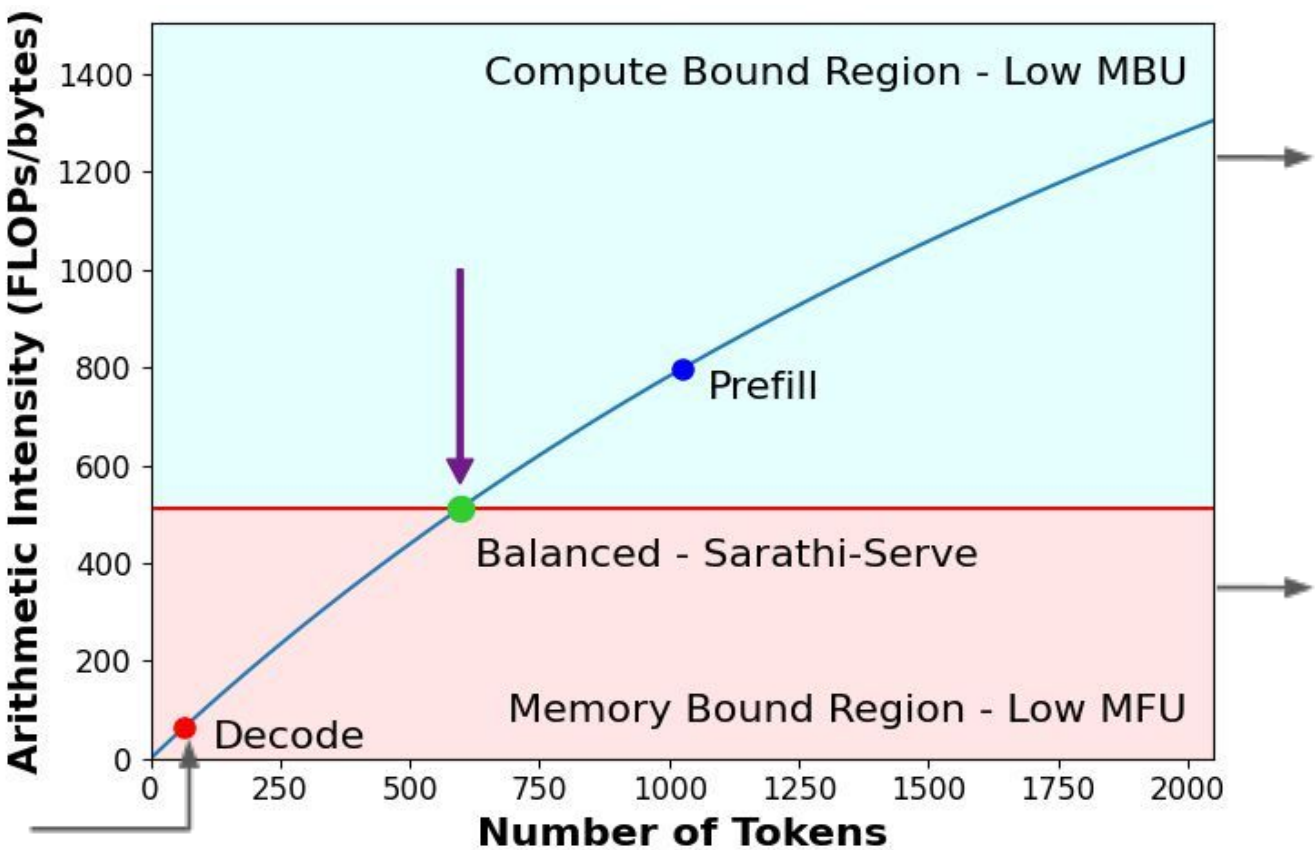


Key Insight

Prefill computation can be done at a marginal cost with careful batching 💡

🔎 Observation: Arithmetic Intensity Slack

Constrained due to memory overhead in decode phase



Latency dominated by compute

Grows linearly with num tokens

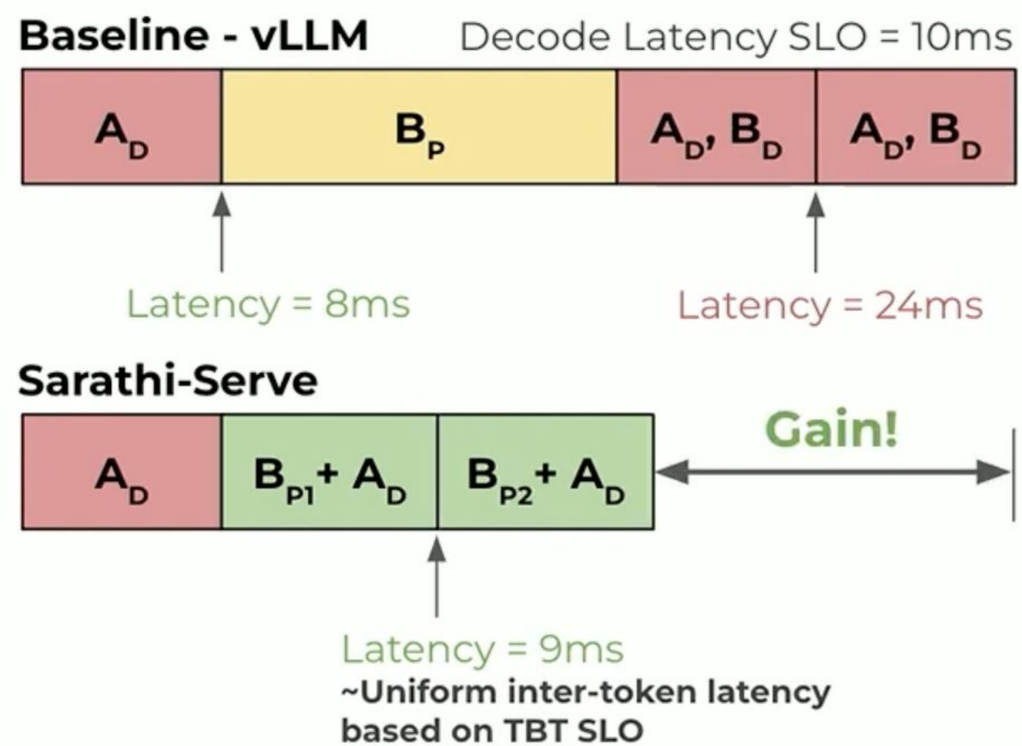
Latency dominated by weight fetch time

Independent of num tokens

🧠 Stall-free Batching

Key Idea

Split large prefills into smaller chunks just enough to consume the leftover compute budget in decode batches



Demo 📺

Evaluations 🧮

Evaluation Setup

Model	Attention Mechanism	GPU Configuration	Memory Total (per-GPU)
Mistral-7B	GQA-SW	1 A100	80GB (80GB)
Yi-34B	GQA	2 A100s (TP2)	160GB (80GB)
LLaMA2-70B	GQA	8 A40s (TP4-PP2)	384GB (48GB)
Falcon-180B	GQA	4 A100s×2 nodes (TP4-PP2)	640GB (80GB)

Model	*relaxed* SLO P99 TBT (s)	*strict* SLO P99 TBT (s)
Mistral-7B	0.5	0.1
Yi-34B	1	0.2
LLaMA2-70B	5	1
Falcon-180B	5	1

Dataset	Prompt Tokens			Output Tokens		
	Median	P90	Std.	Median	P90	Std.
openchat_sharegpt4	1730	5696	2088	415	834	101
arxiv_summarization	7059	12985	3638	208	371	265

📐 Background: Performance Metrics

Time to first token (TTFT): Time required for the first token to show up from the time user submits a request

Time between tokens (TBT): Latency between each output token

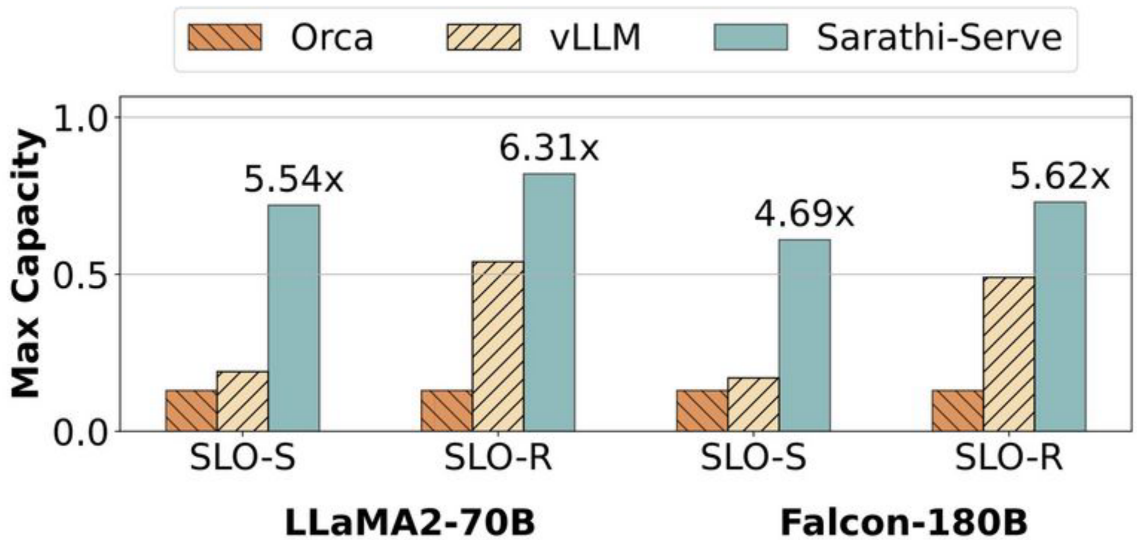
Capacity: Maximum QPS that can be served while satisfying latency SLOs

🧮 Serving Capacity under SLOs

adapt using different chunk sizes

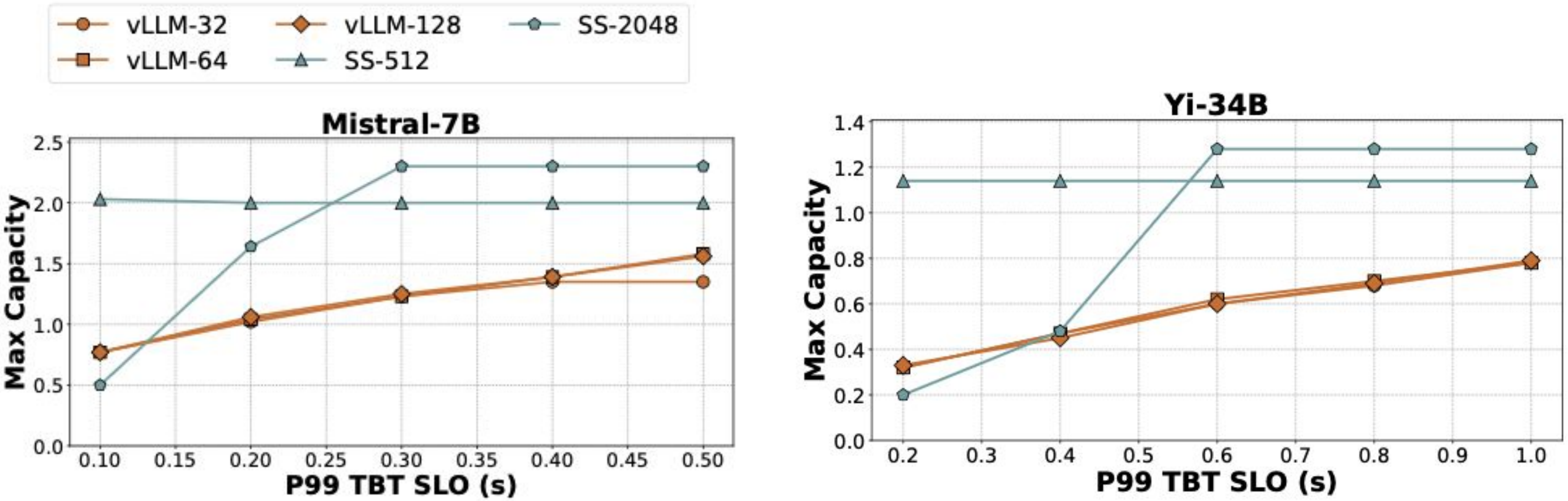
Setup

ShareGPT4 trace on on A100 GPUs with strict (S) and relaxed (R) latency SLOs



5-6x higher capacity

🧮 Throughput-Latency Trade Off



🗒️ Summary

Problem: State-of-the-art systems sacrifice decode latency to achieve higher throughput

💡 **Key Insight** - Low arithmetic intensity of decodes allows for adding compute intensive prefills with negligible decode latency cost

🧮 **Key Results** - We achieve optimality in both latency and throughput simultaneously leading up to 6x higher capacity under SLO constraints

Industry Adoption - Available in all major serving frameworks and more

Paper

GitHub

Use in vLLM