From Tokens to Layers

Redefining Stall-Free Scheduling for MoE Serving with Layered Prefill

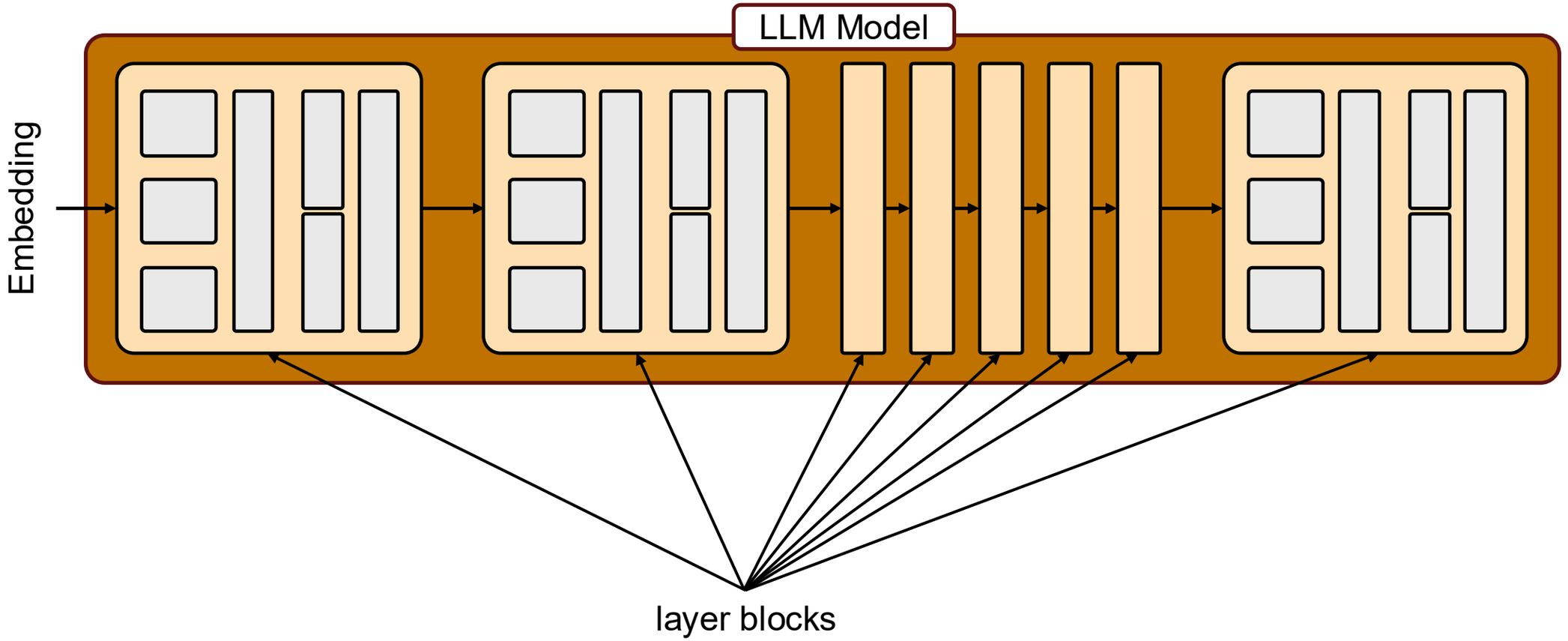
Gunjun Lee, Jiwon Kim, Jaiyoung Park, Younjoo Lee, Jung Ho Ahn

Seoul National University

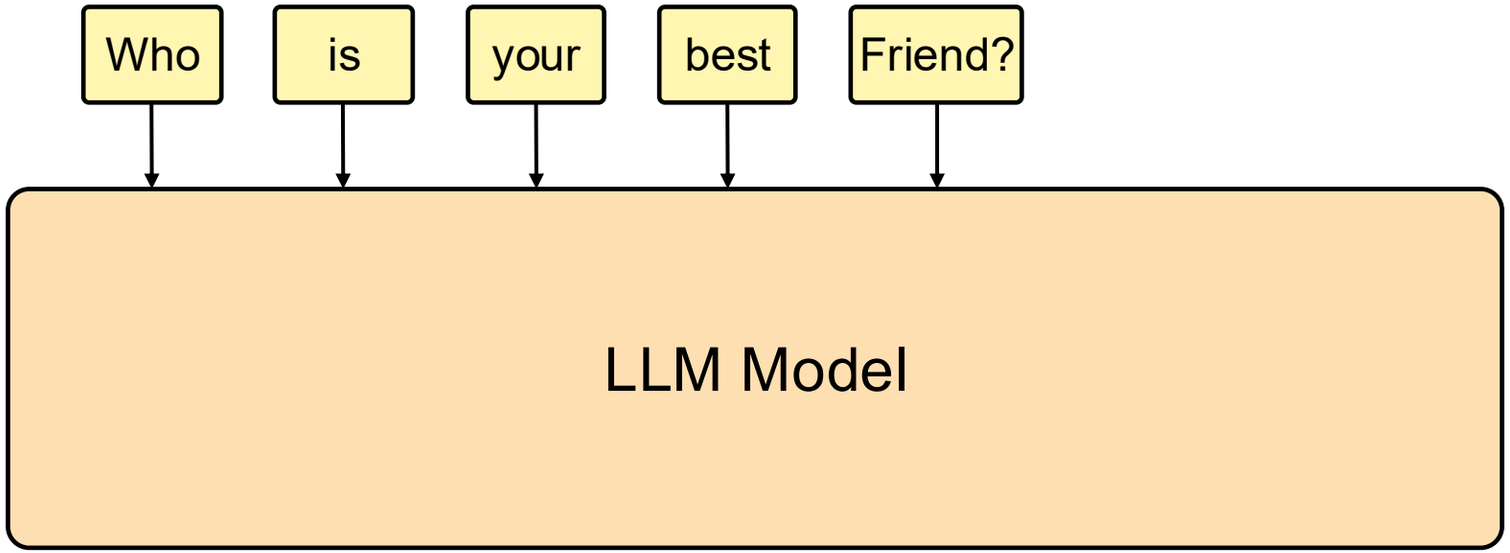
Presenter: Gunjun Lee (kevin970401@snu.ac.kr)

LLM

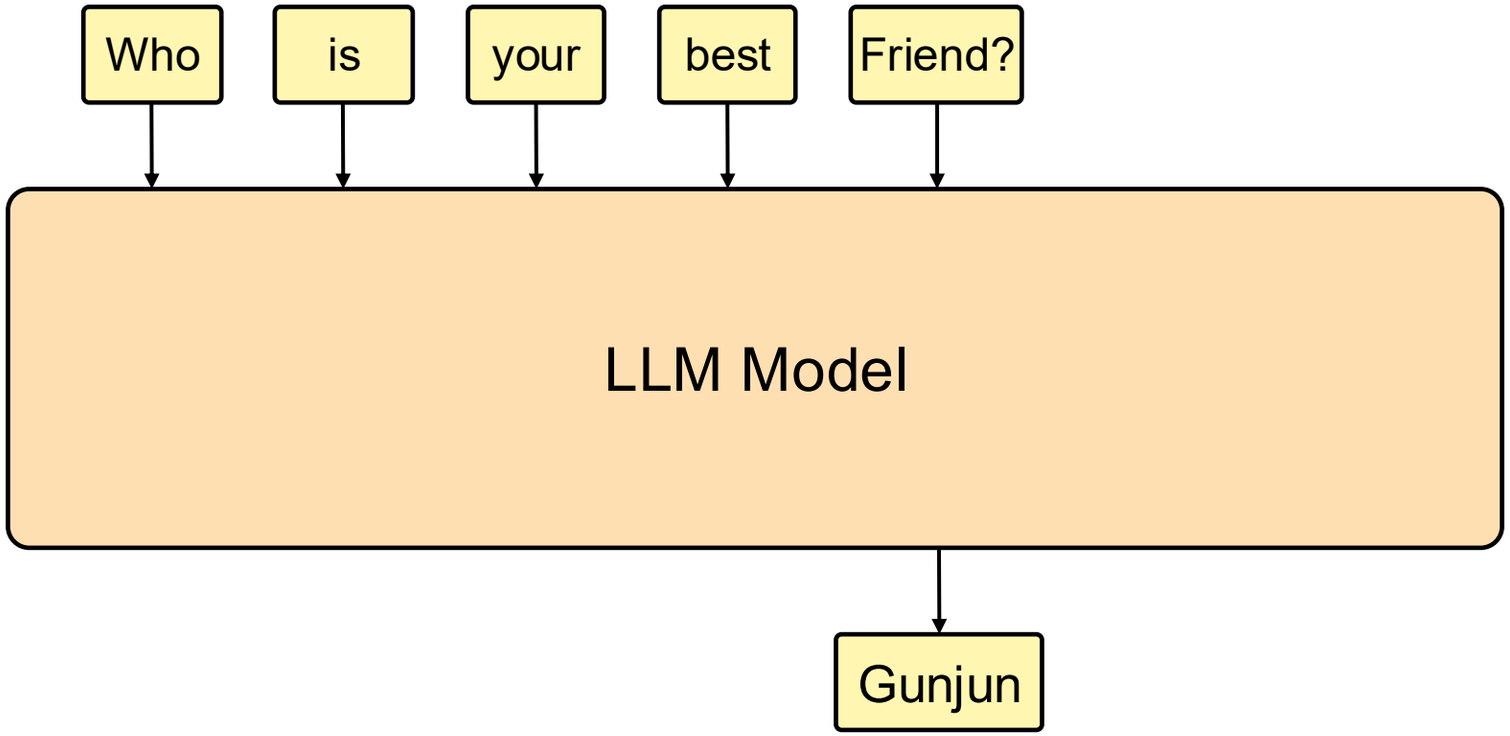
- LLMs consist of repeated layer blocks



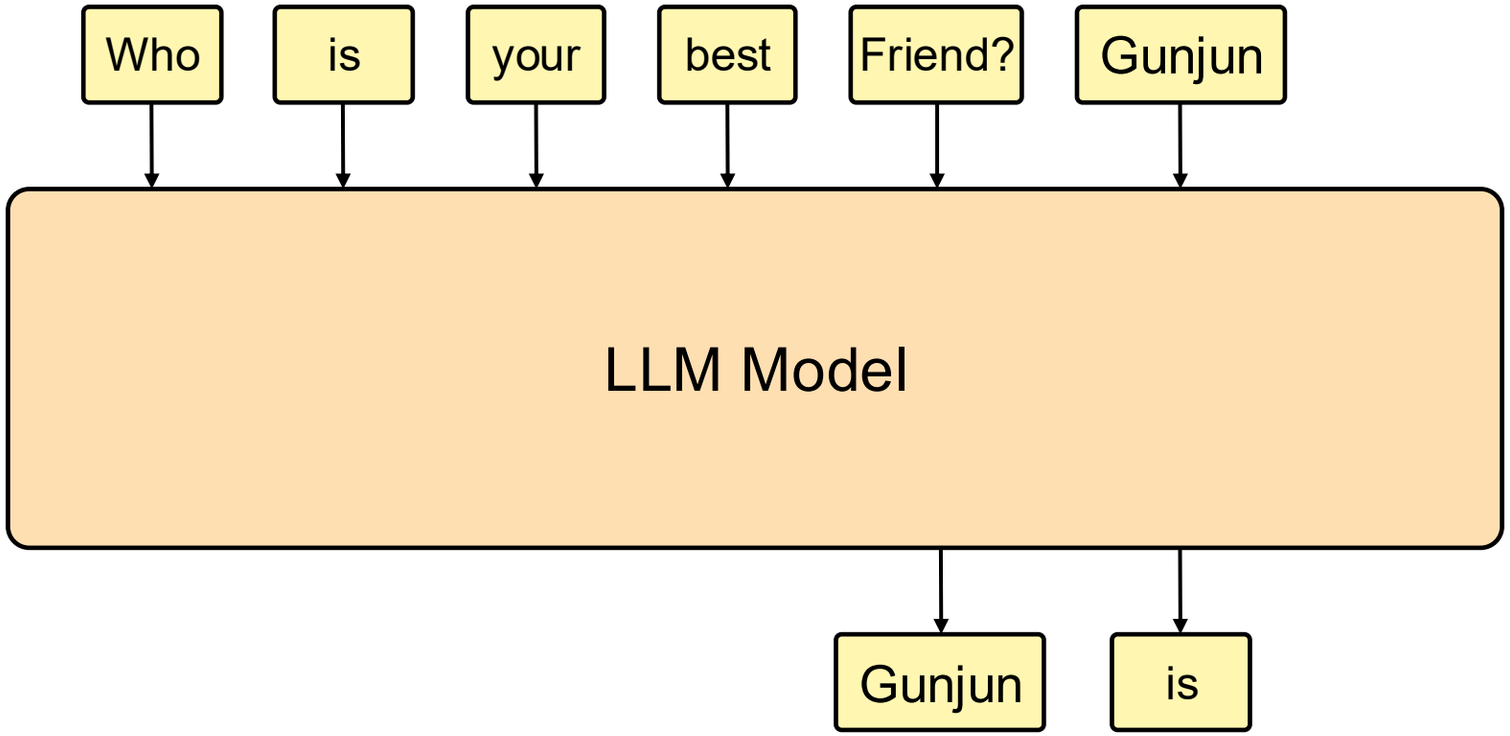
LLM Inference



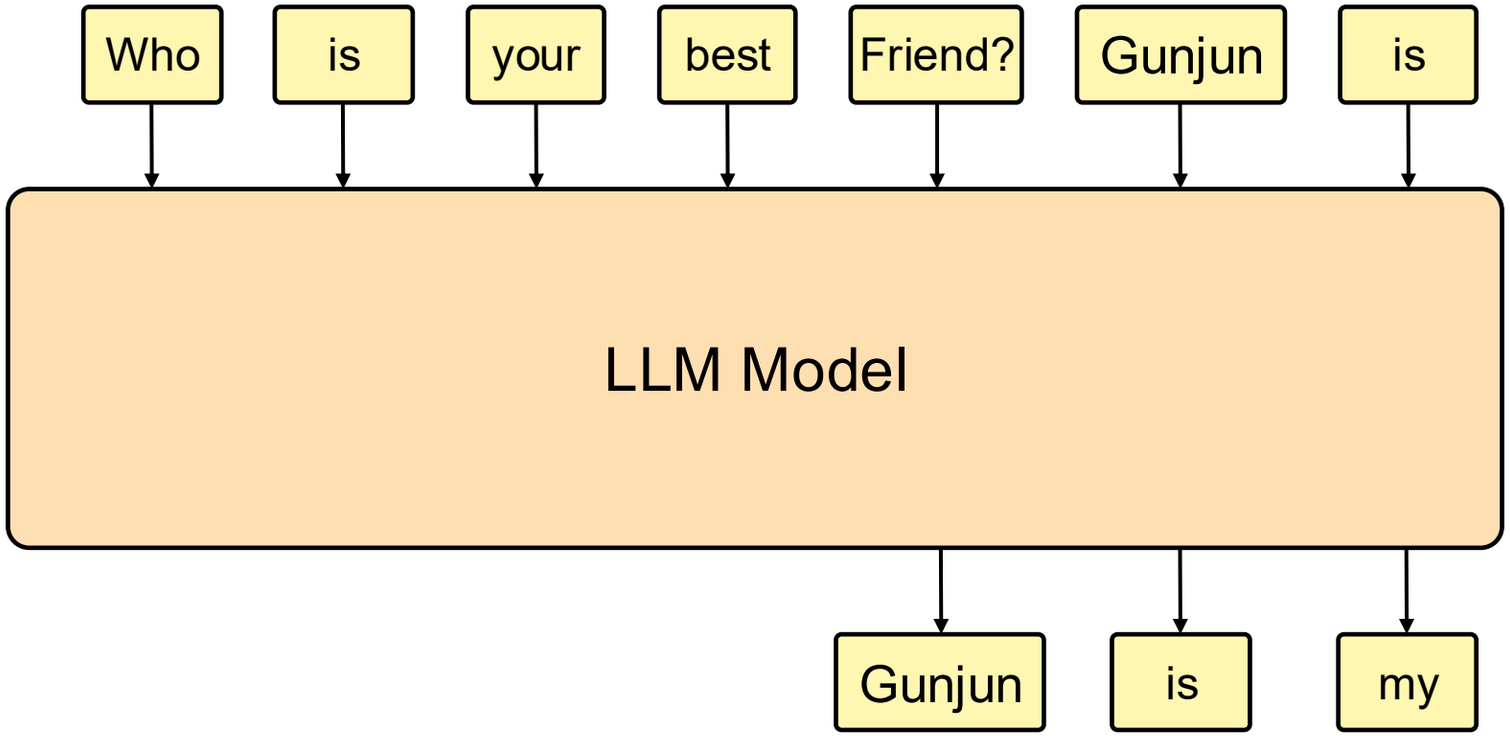
LLM Inference



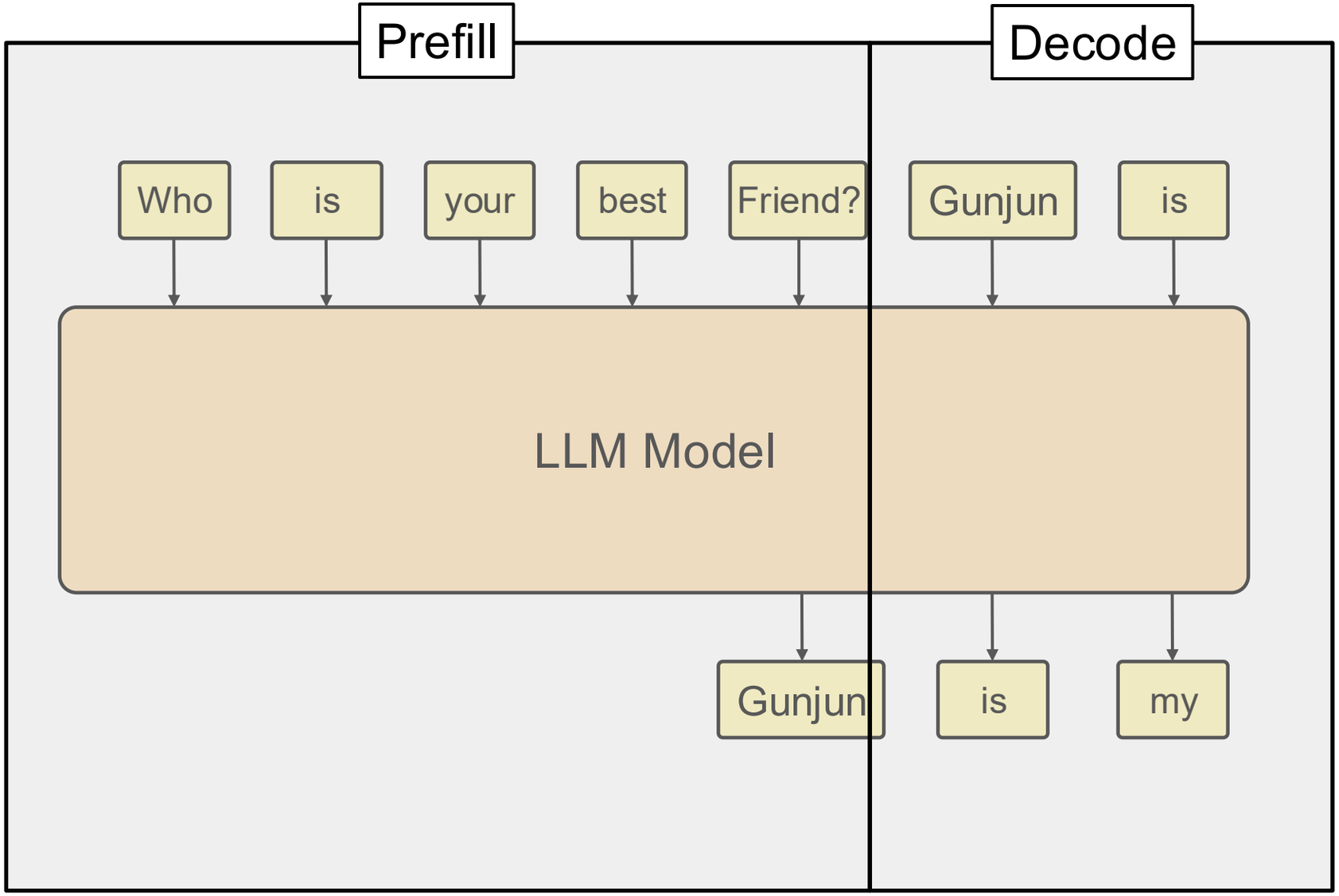
LLM Inference



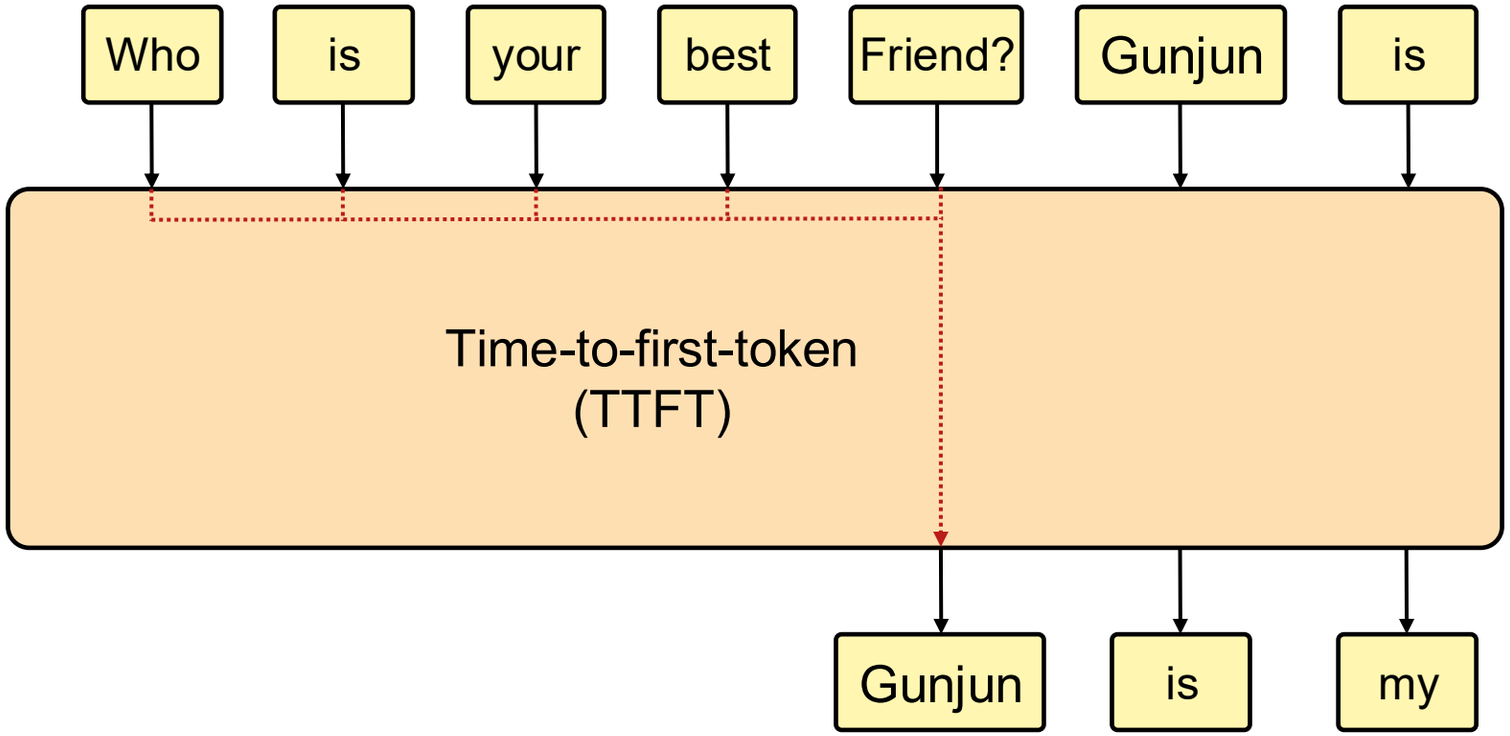
LLM Inference



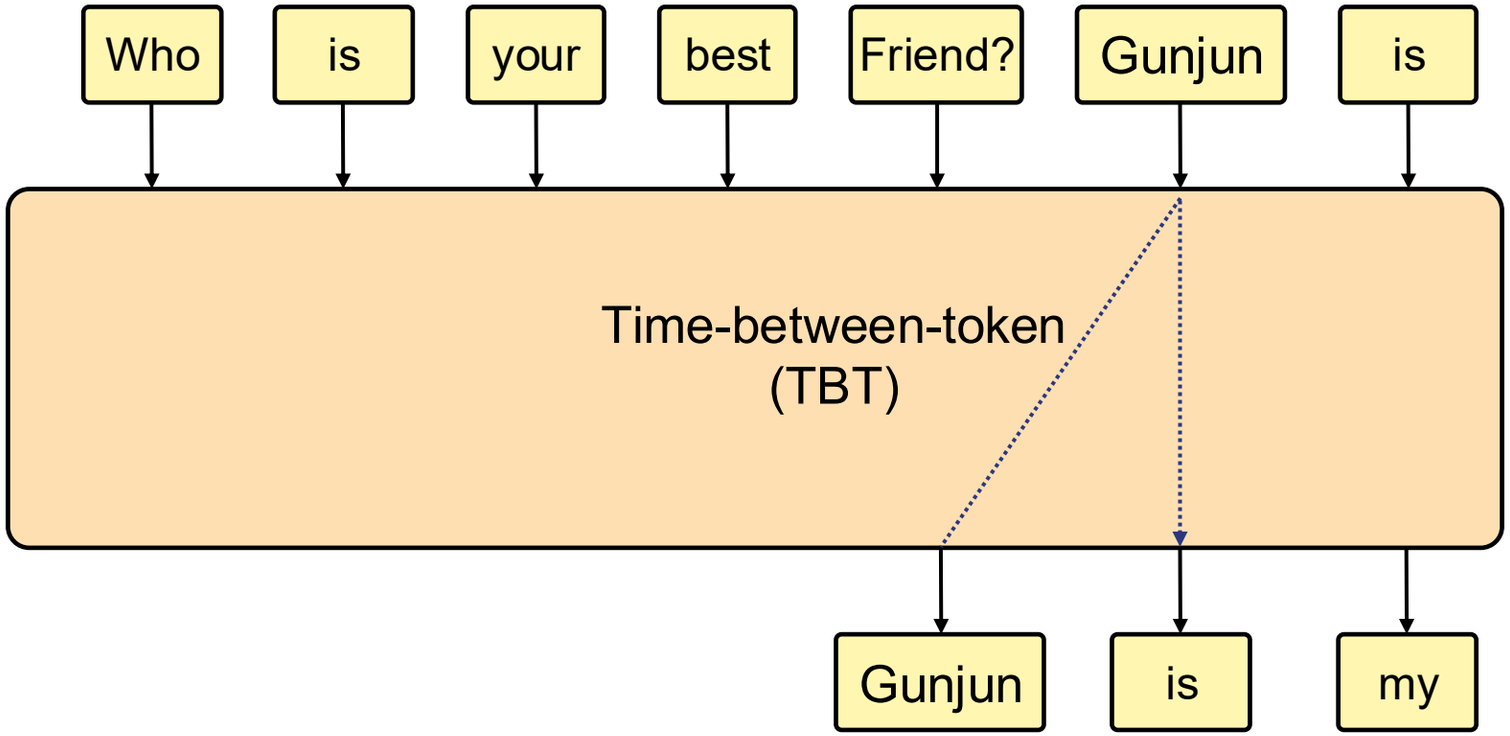
Prefill & Decode



TTFT & TBT

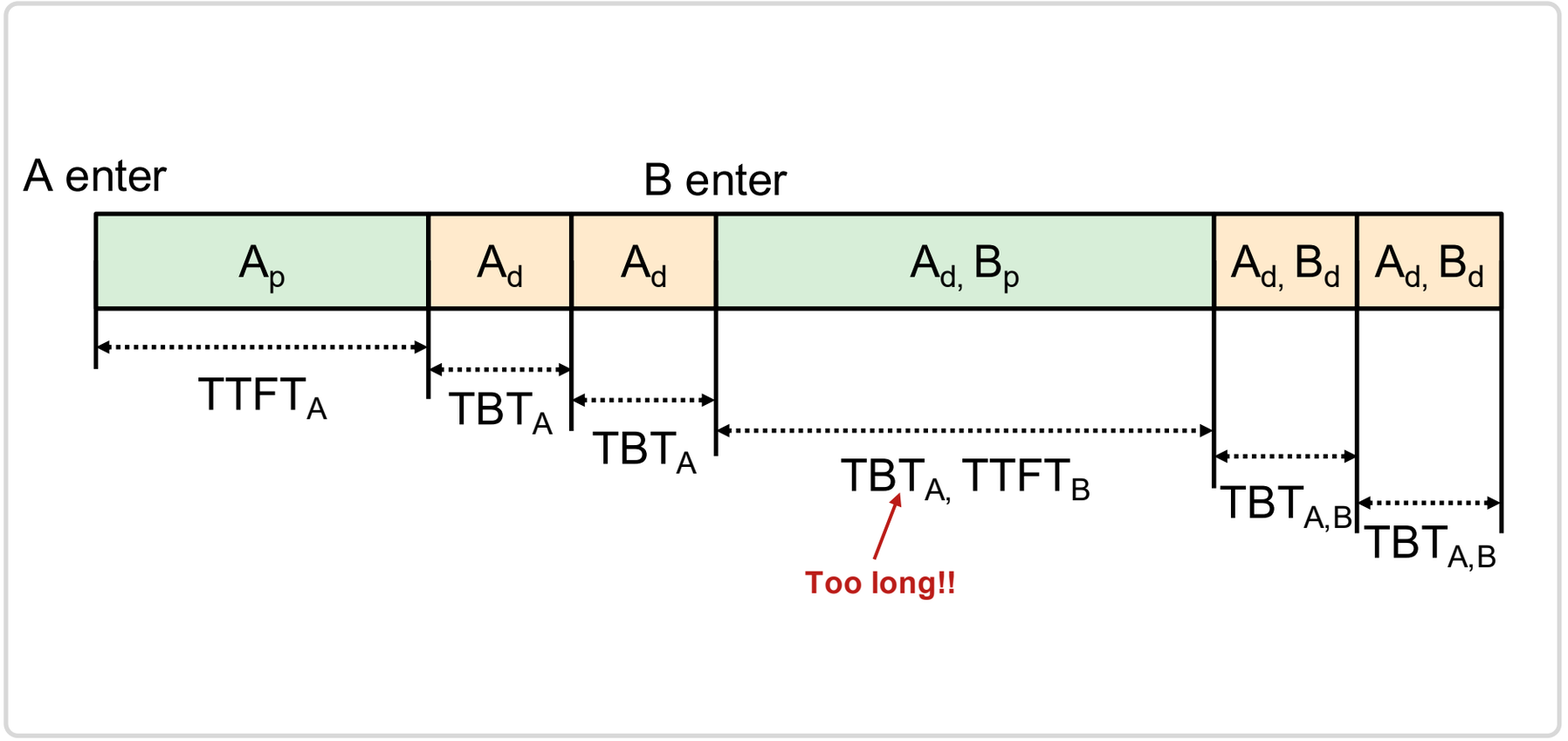

Who
is
your
best
Friend?
Gunjun
is
Time-to-first-token
(TTFT)
Gunjun
is
my

TTFT & TBT


Who
is
your
best
Friend?
Gunjun
is
Time-between-token
(TBT)
Gunjun
is
my

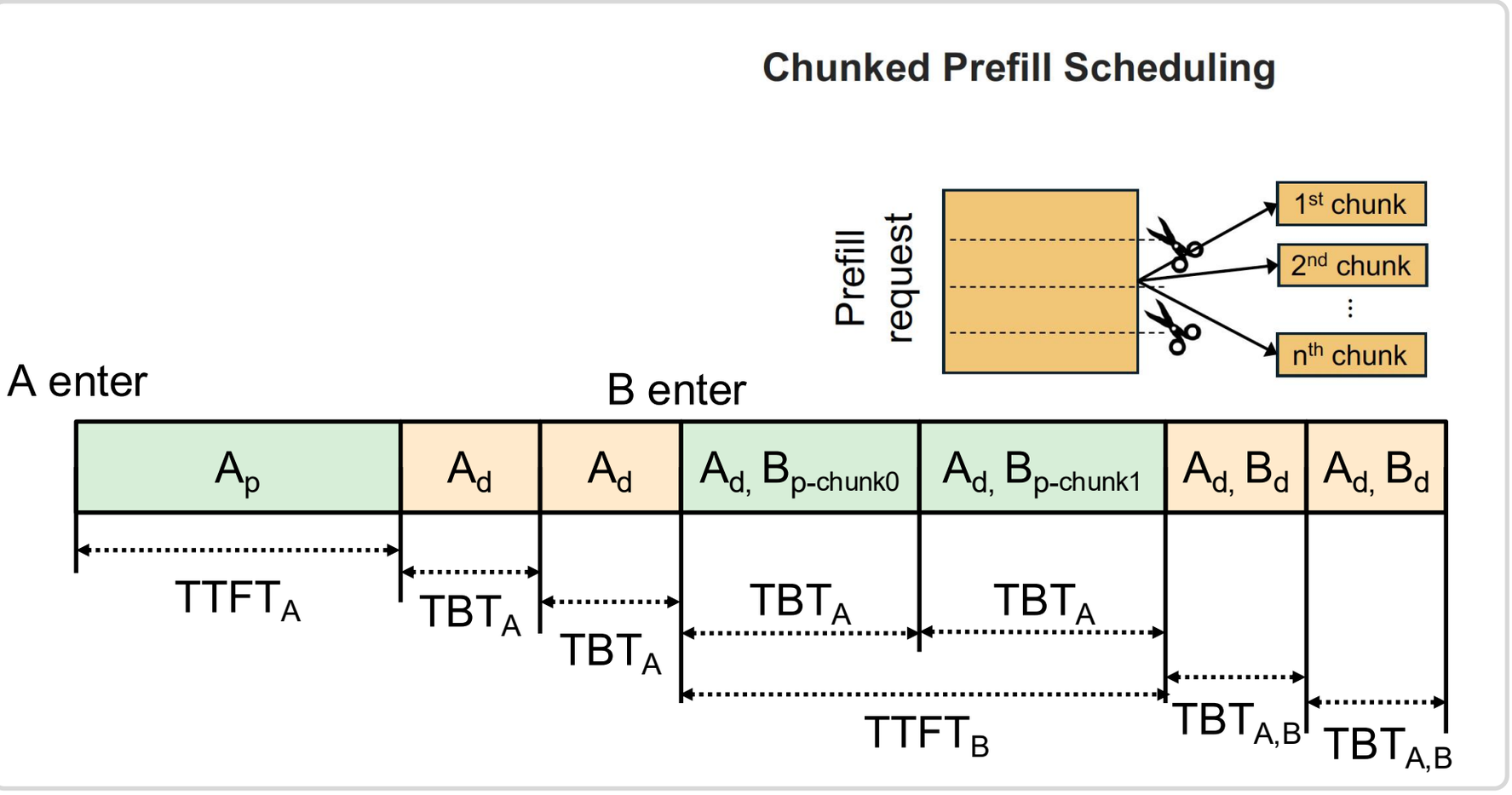
Service level objectives (SLO)

- There are two kinds of SLOs: TTFT, TBT.



Chunked prefill

- Sarathi-Serve[1] proposed *chunked prefill* to make decoding stall-free by slicing the prompt along tokens.

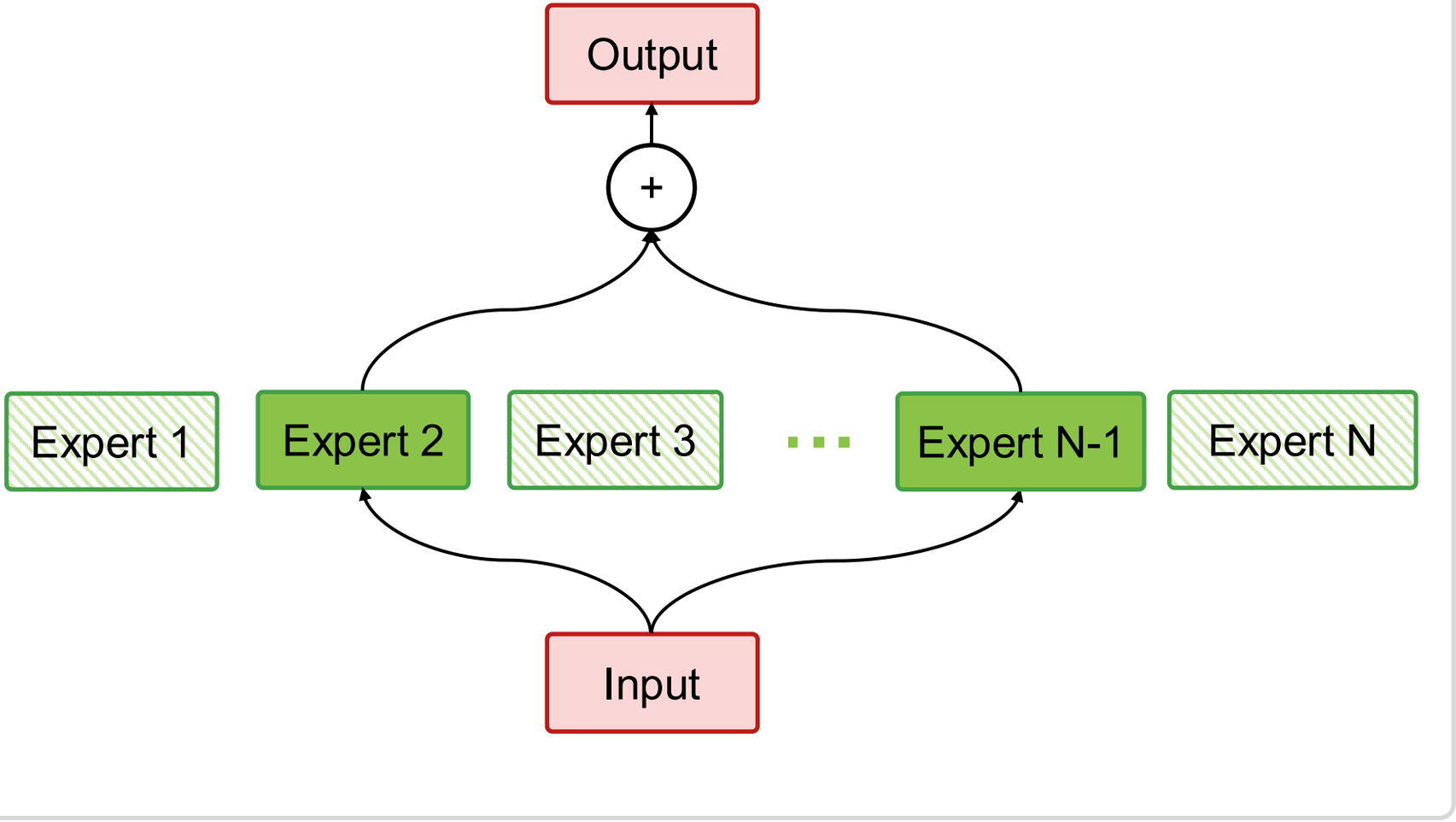


- Small chunks protect TBT by limiting the prefill work inserted into each iteration.

[1] Agrawal Amey et al. “Taming Throughput-Latency Tradeoff in LLM Inference with Sarathi-Serve.” OSDI `24.

Mixture-of-Experts (MoE)

- MoE[1] has emerged as the dominant strategy for scaling.



[1] Noam Shazeer et al. “Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer.” ICLR `17.

Sparsity Erosion problem

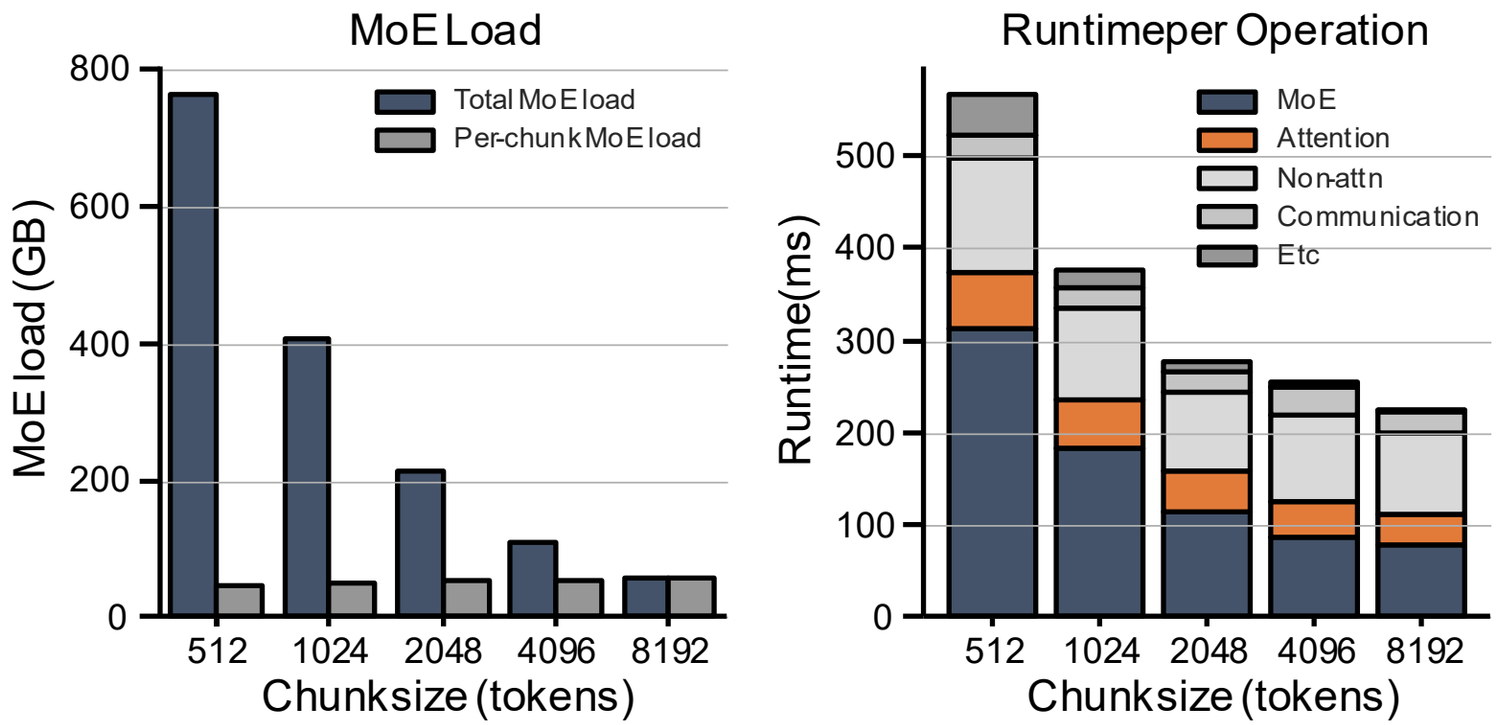
- Suppose the chunk size = 256, top-k = 8, # of experts = 128
  - Only 16 tokens are routed to each expert
  - 16 is far below the ridge point of modern AI accelerators.
  - We formulate this issue as a sparsity erosion problem.

Table 1. Expert weight coverage ratio as a function of decode batch size, measured on `Qwen` with the ShareGPT dataset.

Batch Size	1	2	4	8	16	32	64	128	256	≥512
Coverage (%)	6.25	11.7	21.3	29.0	44.5	54.7	69.4	86.3	93.4	≥98

Chunked prefill implication

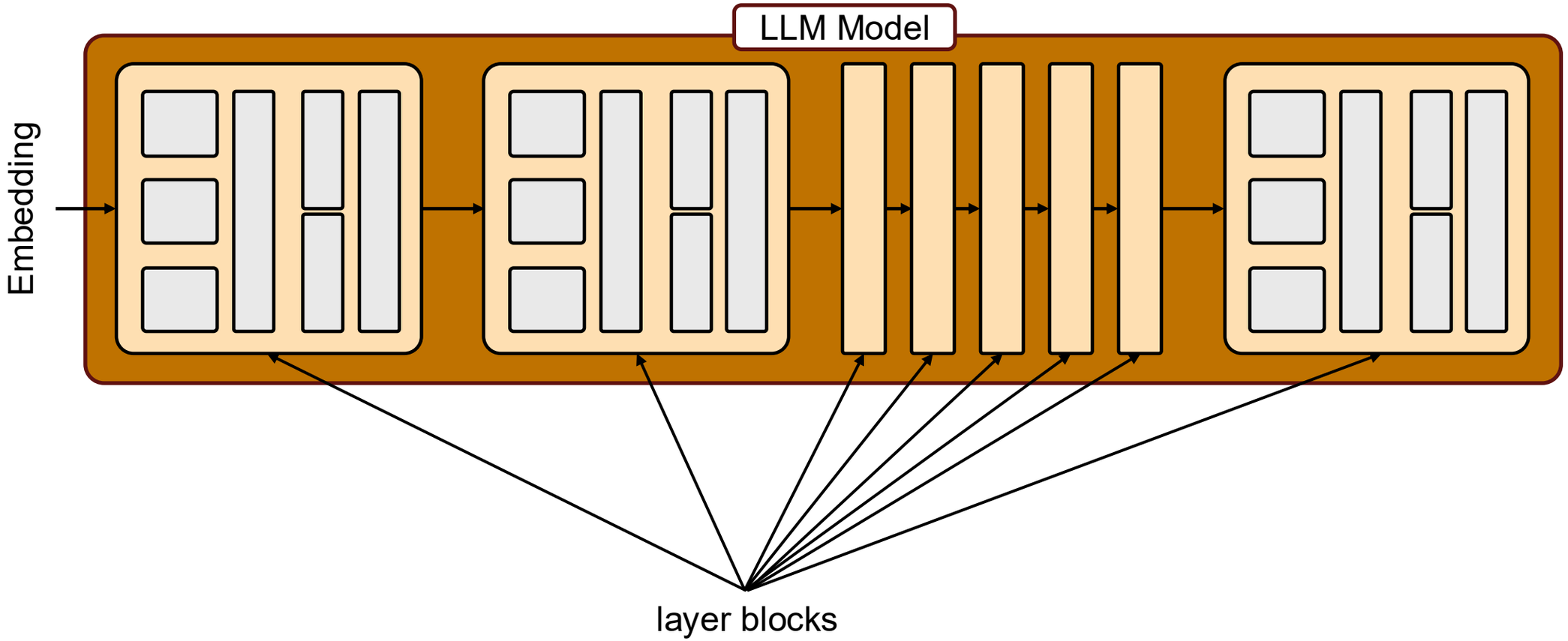
- Chunked prefill causes redundant memory loading in MoE models.



- With a fixed 8192 token input, total MoE load falls sharply as chunk size increases.
- At 512-token chunks, MoE accounts for more than half of prefill runtime
- Larger chunks improve reuse but raise iteration duration, which can violate TBT SLOs.

LLM

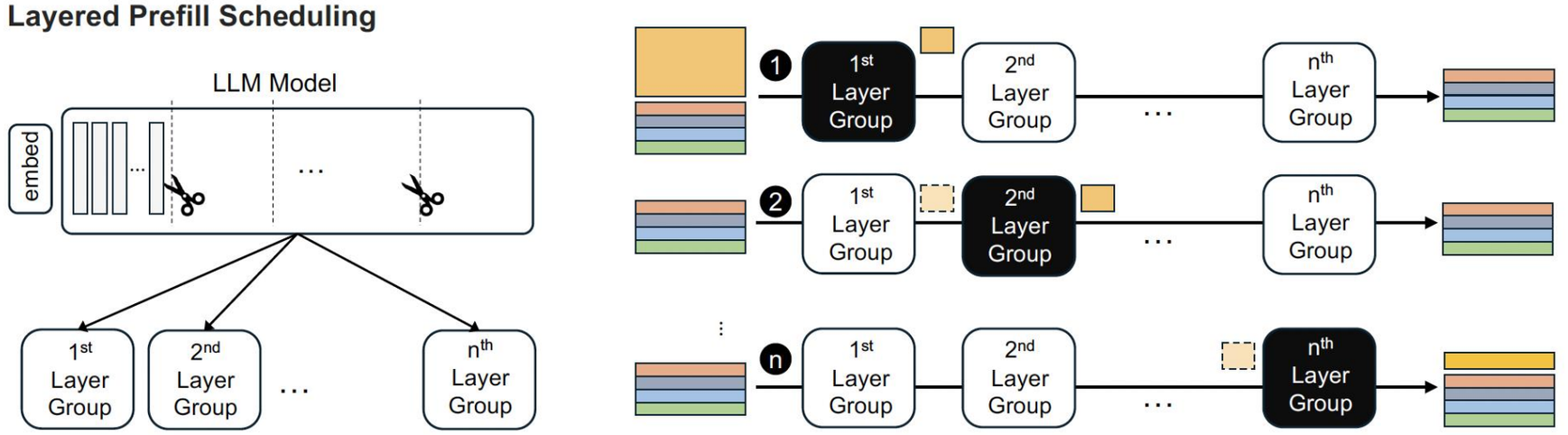
- LLMs consist of **repeated** layer blocks



How about slicing the LLM model along the layer axis?

Layered prefill

- Only one group performs prefill & decode in each iteration.



- Prefill completes after N_{lg} iterations.
- Each layer traverses the prompt once.
- MoE expert reloads across token chunks are eliminated.

The same stall-free decode property is preserved, but redundant MoE weight traffic is removed.

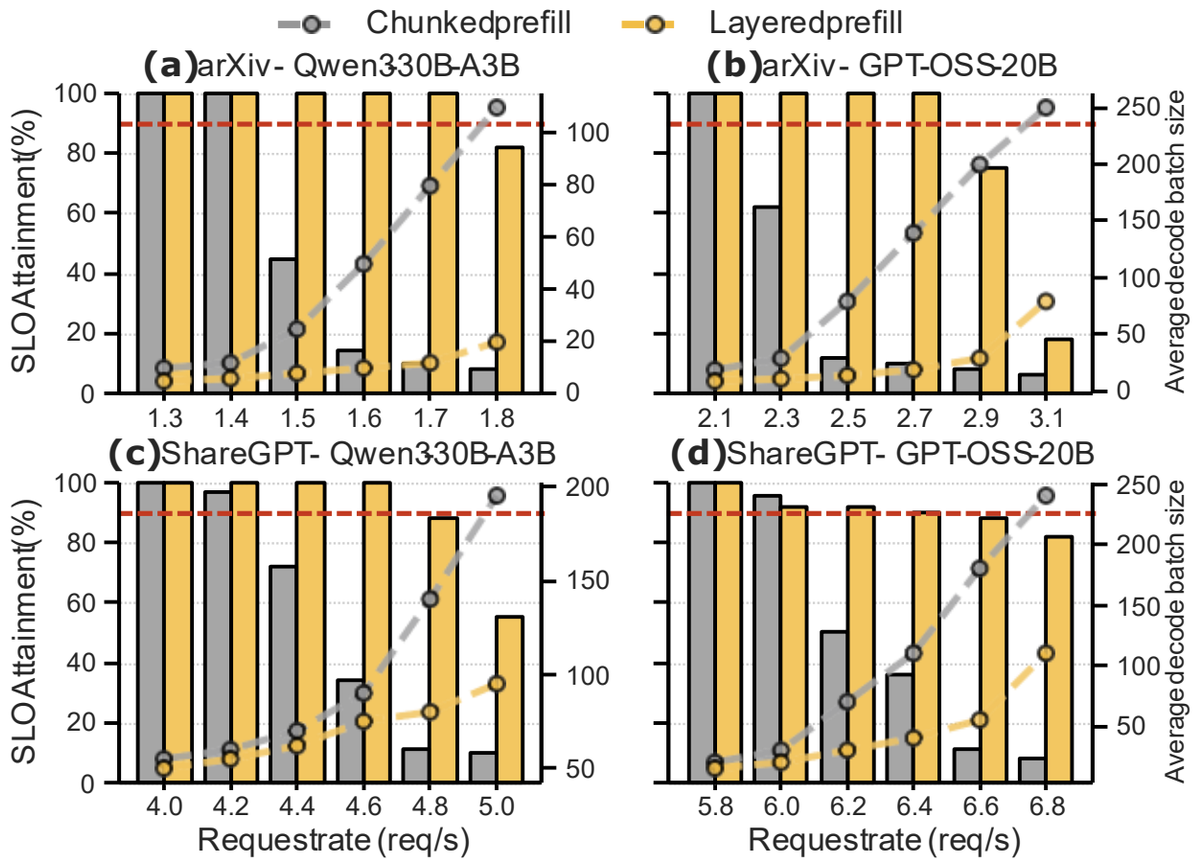
Evaluation Setup

- Experiments use MoE models with different routing configurations on H100 GPUs

Category	Setup
Hardware	2x NVIDIA H100 80GB GPUs with NVLink
Models	Qwen3-30B-A3B: 128 experts, top-8 routing GPT-OSS-20B: 32 experts, top-4 routing
Datasets	ShareGPT: conversational prompts, mean input length 2,340 tokens arXiv Summarization: long prompts, mean input length 9,194 tokens
SLOs	TTFT: 5 s for ShareGPT, 10 s for arXiv TBT: 125 ms for Qwen, 100 ms for GPT
Metrics	TTFT, TBT SLO attainment, energy per output token

SLO attainment

- Layered prefill sustains SLO attainment at higher request rates.
  - The advantage appears both models and both workloads.

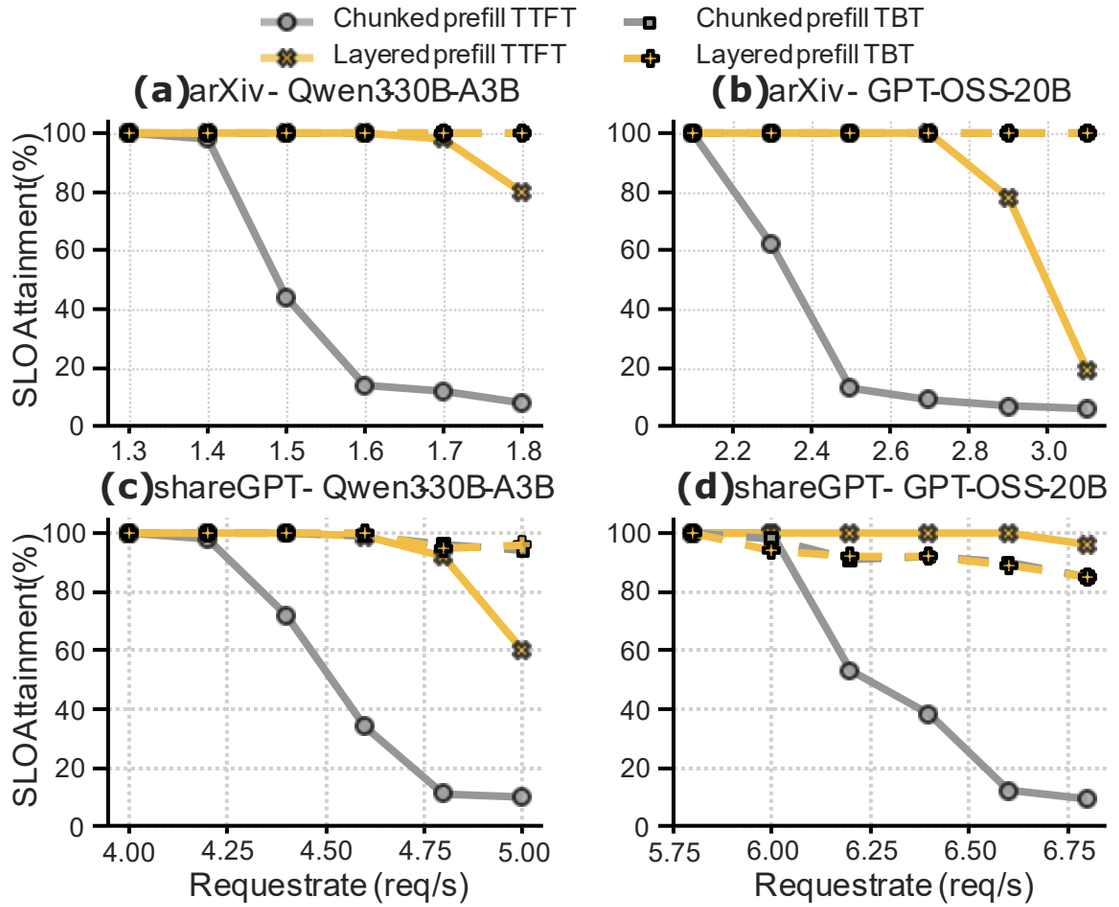


- Qwen on arXiv: layered prefill stays near 100% through 1.7 req/s, while chunked prefill collapses at 1.5 req/s.
- GPT on arXiv: layered prefill maintains 100% through 2.7 req/s, while chunked prefill collapses at 2.3 req/s.

The gap is larger on arXiv because long prompts create more token chunks, making chunk-induced MoE reloads more severe.

SLO breakdown

- Both schedulers keep TBT near-perfect over most of the operating range.



- Layered prefill reduces queuing pressure and prefill runtime, so TTFT attainment declines much later as load rises.

TBT remains stable because every iteration still includes decode work across all layer groups.

Energy also improves

- The energy gain holds at equal request rate and at higher sustainable request rates.

22%
Lower energy per token for Qwen at higher usable throughput

20%
Lower energy per token for GPT at higher usable throughput

23-29%
Higher sustainable request rate under the same SLO envelope.

70%+
Mean TTFT reduction at the same request rate for GPT

Model	Method	Req/s	Mean TTFT (s)	P99 TTFT (s)	Mean TBT (ms)	Energy (mJ/tok)
Qwen	Chunked	1.3	2.80	8.65	32.2	56.6
Qwen	Layered	1.6	2.46	8.01	28.1	44.2
GPT	Chunked	2.1	3.00	8.57	25.1	37.4
GPT	Layered	2.7	2.18	6.86	25.7	29.8

Conclusion

- Layered prefill opens a better operating point for MoE serving.

70%	41%	39%	22%
lower TTFT	lower end-to-end latency	lower expert-load traffic	lower energy per token

Thank you!

Backup Slides

Experiments with various configurations

Table 9. Broader validation across GPUs, models, and workloads.

GPU	Model	Method	TTFT (s)		TBT (ms)	
			Mean	p99	Mean	p99
A100x2	Qwen3-30B-A3B (arXiv, 0.5 req/s)	Chunked (512)	1.32	5.40	22.4	70.4
		Layered ($N_{lg} = 16$)	0.962	3.81	19.3	53.6
H100x2	gpt-oss-120b (fp4) (arXiv, 1.2 req/s)	Chunked (512)	3.11	8.82	34.9	47.2
		Layered ($N_{lg} = 12$)	1.09	3.89	20.5	44.8
H100x8	Qwen3-235B-A22B (arXiv, 1.2 req/s)	Chunked (512)	4.25	11.6	41.7	50.5
		Layered ($N_{lg} = 16$)	2.26	7.42	33.1	45.0
H100x8	Qwen3-235B-A23B (ShareGPT, 3.5 req/s)	Chunked (512)	6.65	16.4	50.2	70.2
		Layered ($N_{lg} = 16$)	3.35	8.48	50.8	88.5
H100x8	gpt-oss-120b (fp4) (arXiv, 3.0 req/s)	Chunked (512)	1.05	3.45	14.4	18.6
		Layered ($N_{lg} = 12$)	0.530	1.75	11.8	19.3
H100x8	gpt-oss-120b (fp4) (ShareGPT, 6.5 req/s)	Chunked (512)	0.212	0.791	12.2	19.6
		Layered ($N_{lg} = 12$)	0.178	0.572	11.3	21.0