Chunked Prefill

INTRODUCTION TO LLM INFERENCE SERVING SYSTEMS

CHUHONG YUAN

Late Submission Policy

- Only 3 late submissions and 2 missed submissions (included in the 3) are accepted
- Incomplete submissions counted as 0.5 of 3 after submitting a complete one

Homework Review

- Most submissions are great
- Missing point: logical connections between design and motivation, evaluation and design
- Why is the logical connection important?
    - The backbone of a paper
    - Convince the readers of the significance of the work
    - Basis of critical thinking

Background – Latency vs. Throughput

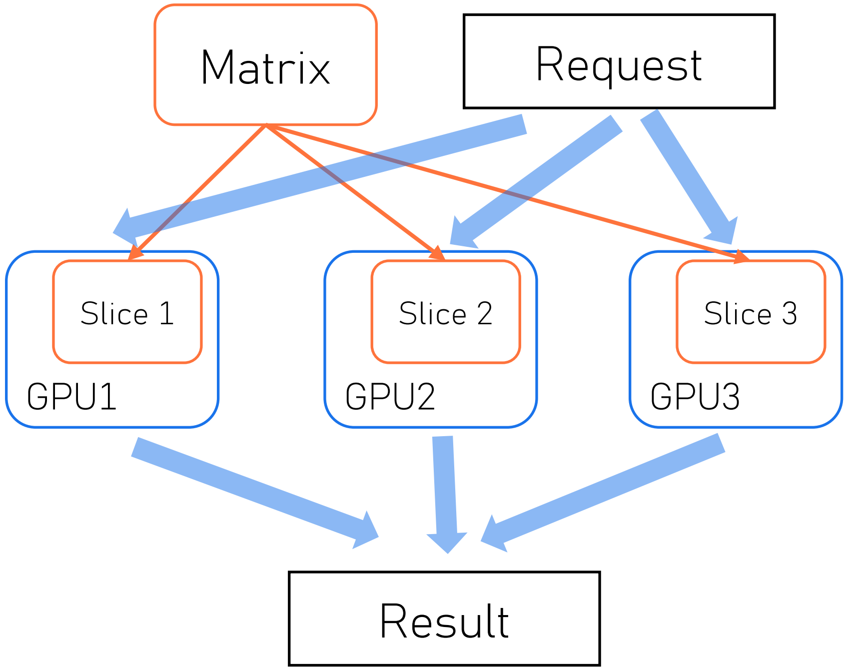
- Decoding is memory-bound and has lower computation utilization rate
- Batching strategy for decoding
  - Smaller batches – smaller TPOT but smaller throughput
  - Larger batches – larger throughput but larger TPOT

Background – Parallelism

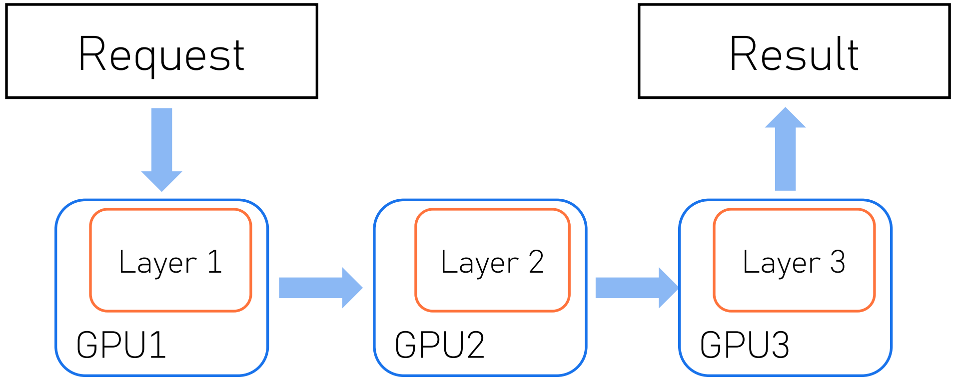
- Tensor parallelism
- Pipeline parallelism
- Data parallelism
- Expert parallelism

Background – Parallelism

- Tensor parallelism
  - Split layers among different GPUs
  - Need to merge the outputs
- Pipeline parallelism
- Data parallelism
- Expert parallelism



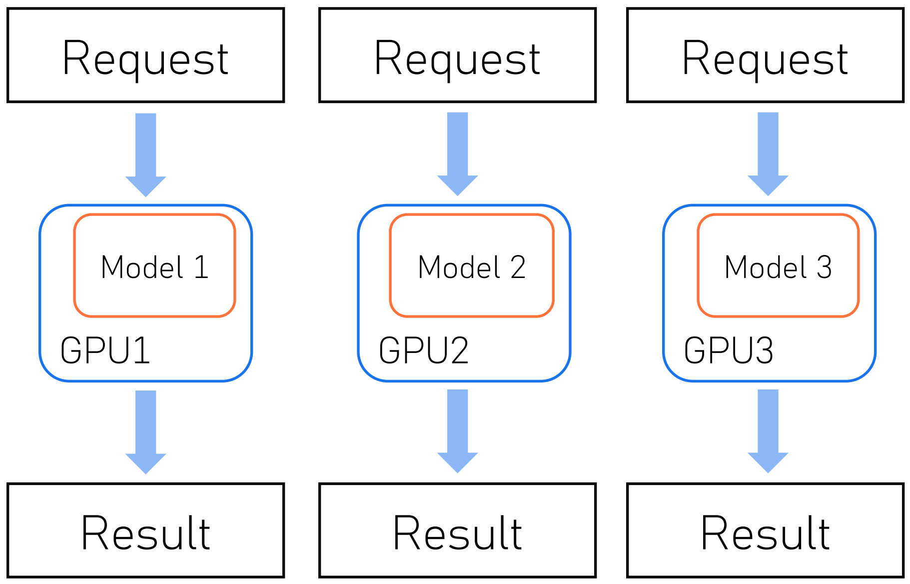
Background – Parallelism



- Tensor parallelism
- Pipeline parallelism
    - Split the layers into stages
    - Deploy to GPUs
    - A request needs to go through a pipeline of GPUs to finish
- Data parallelism
- Expert parallelism

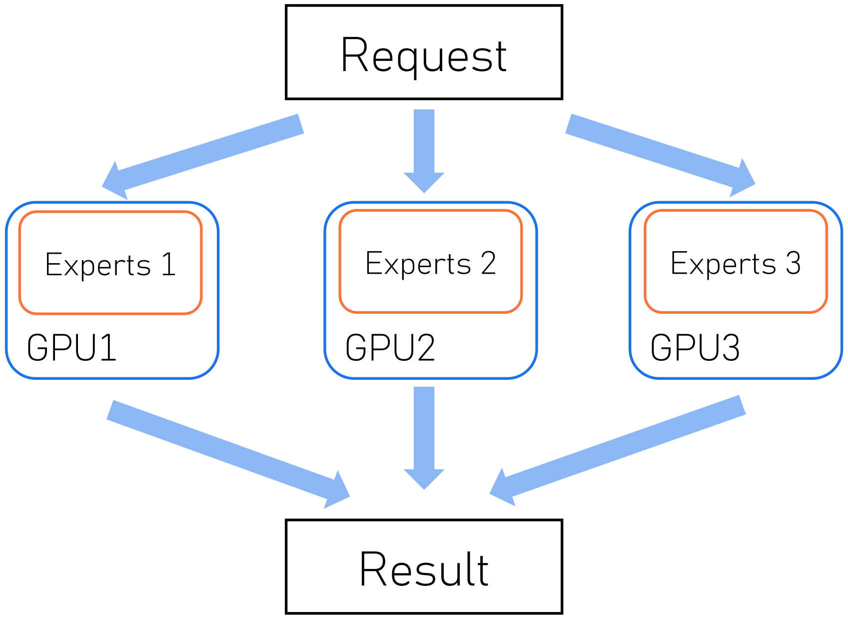
Background – Parallelism

- Tensor parallelism
- Pipeline parallelism
- Data parallelism
  - Deploy multiple model replicas
  - Route the requests to different replicas
- Expert parallelism

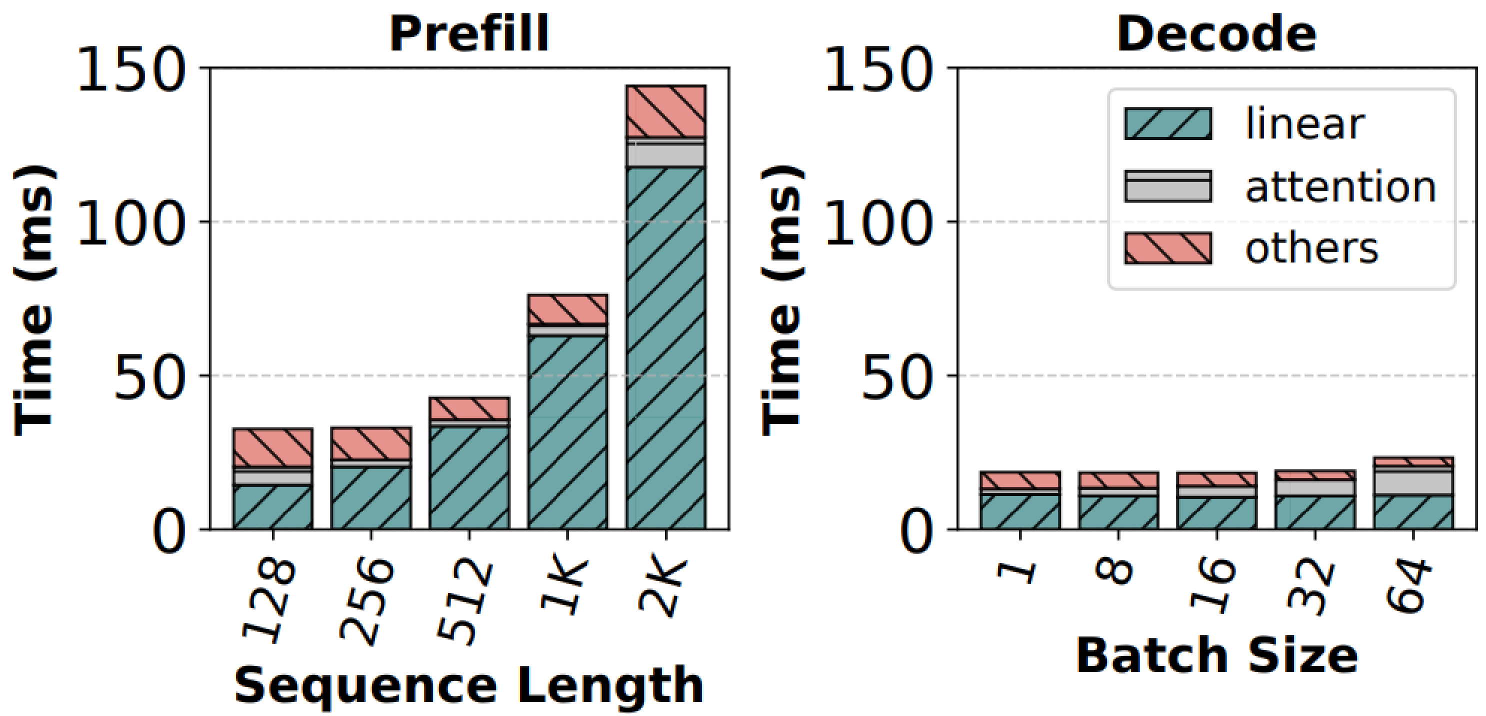


Background – Parallelism

- Tensor parallelism
- Pipeline parallelism
- Data parallelism
- Expert parallelism
  - For Mixture-of-Experts models
  - Each GPU holds parts of experts

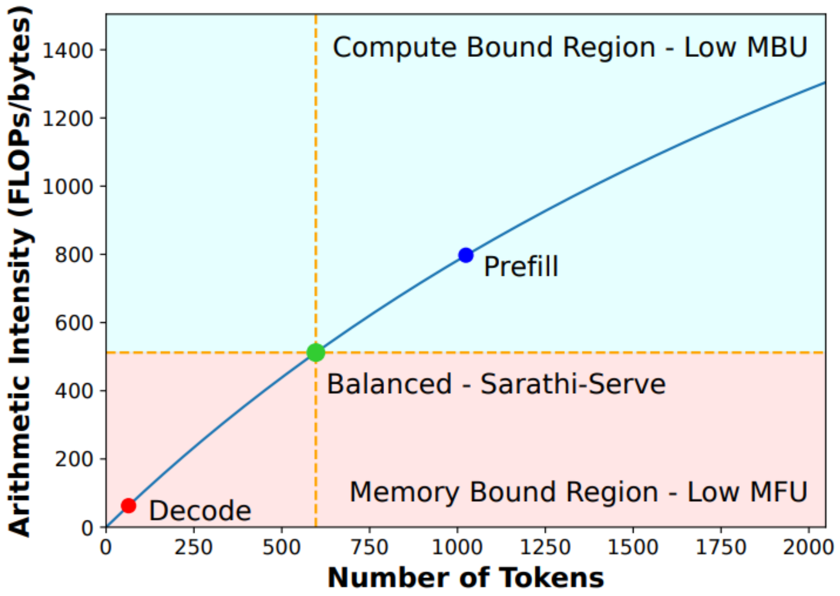


Background – Cost of Prefill & Decoding



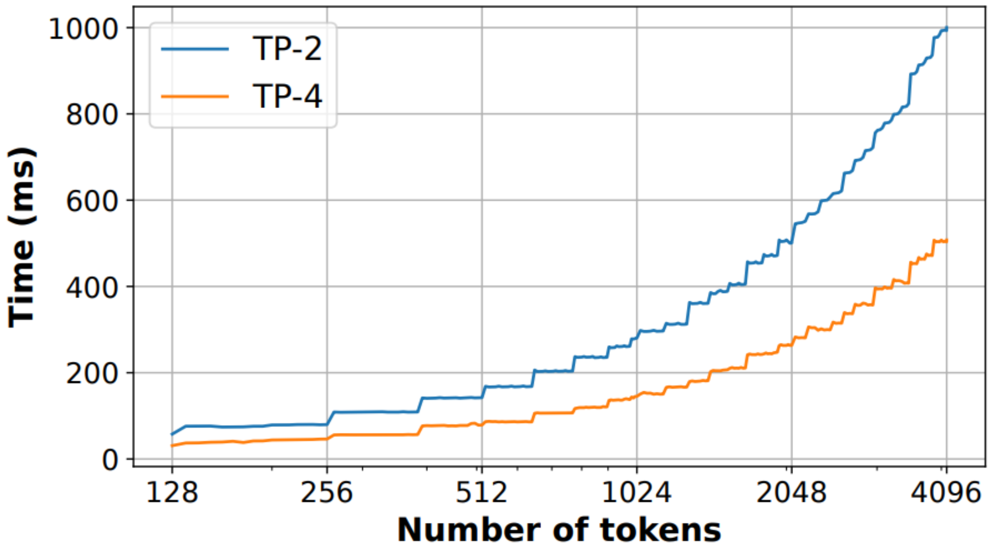
Background – Cost of Prefill & Decoding

- Decoding computation, if only one task one time, wastes resources
- Execution time: $\max(T_{math}, T_{mem})$
- When $T_{math} = T_{mem}$, both utilization are maximized



Background – Cost of Prefill & Decoding

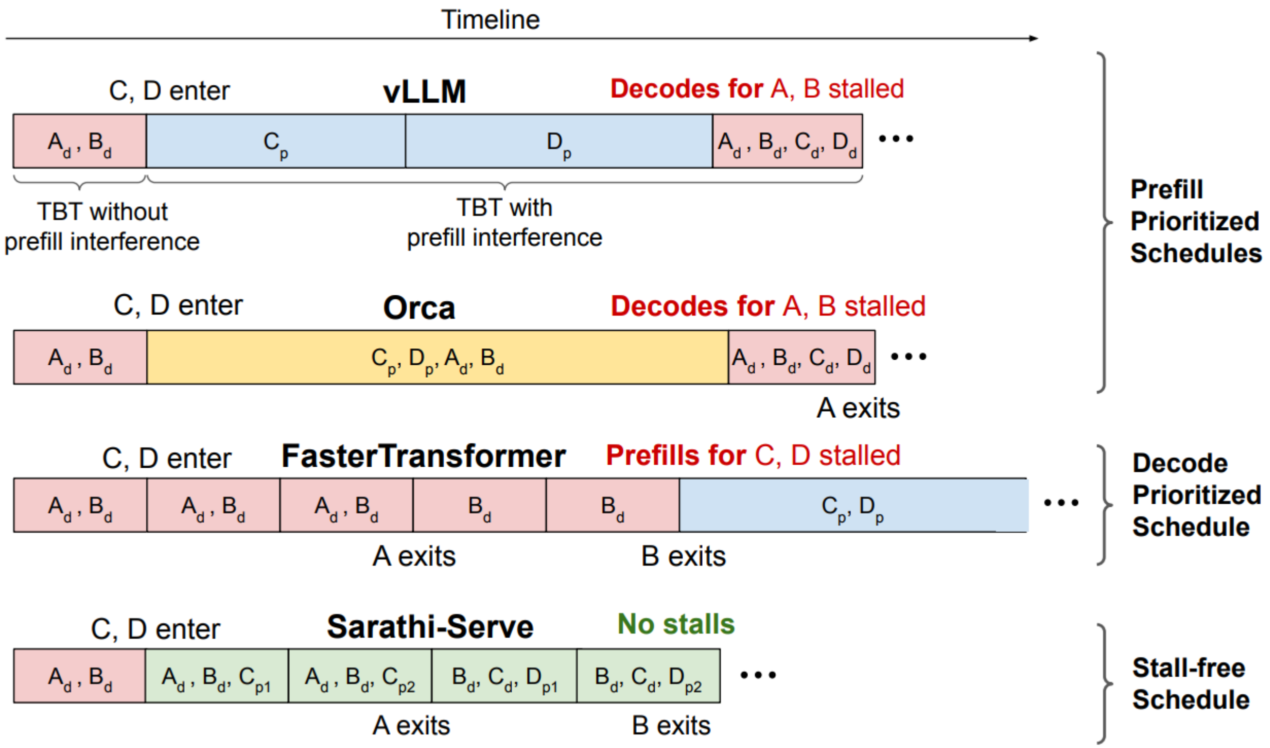
- Decoding computation, if only one task one time, wastes resources
- Execution time: $\max(T_{math}, T_{mem})$
- When $T_{math} = T_{mem}$, both utilization are maximized



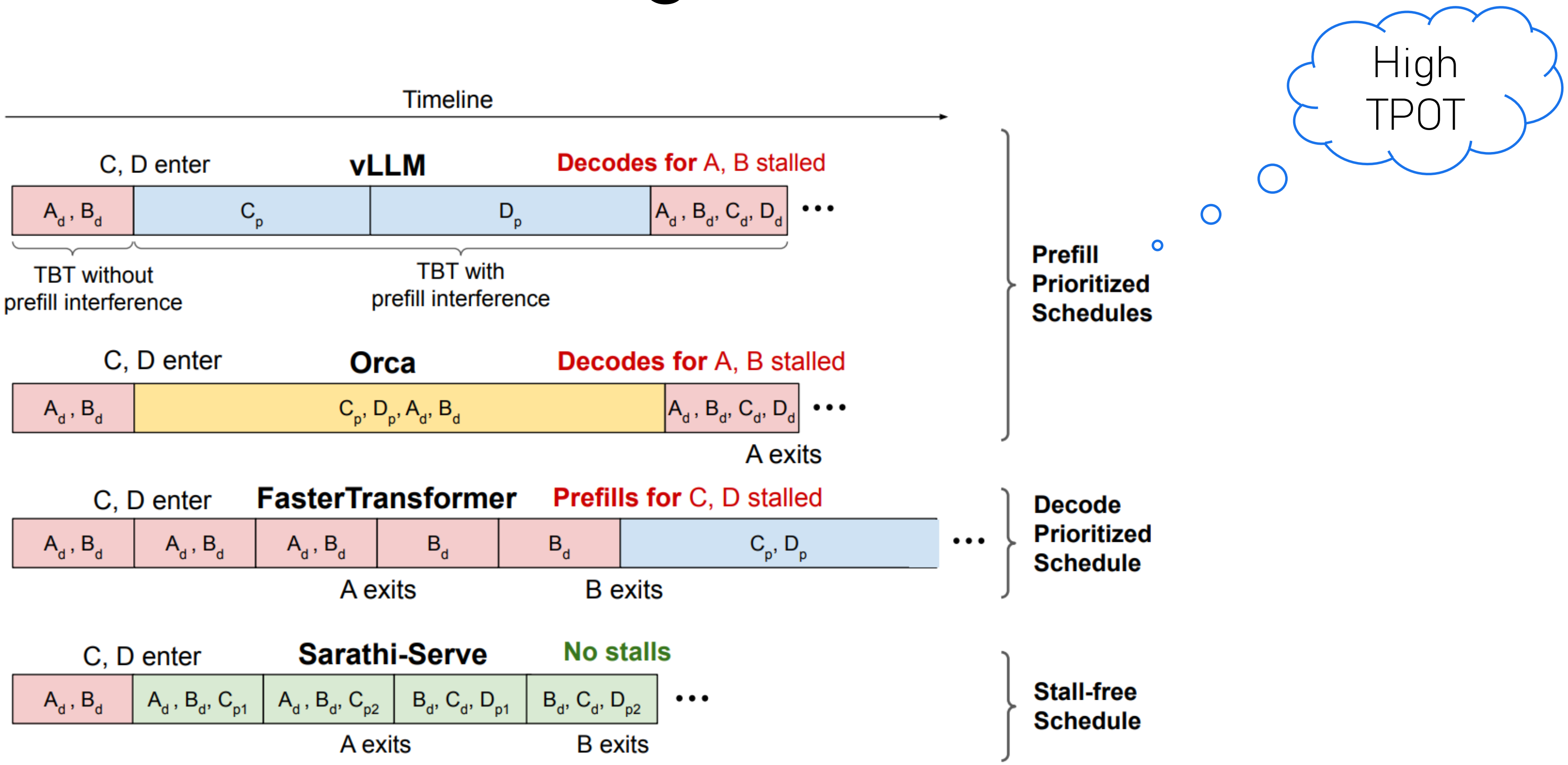
Background – Cost of Prefill & Decoding

- Decoding computation, if only one task one time, wastes resources
- Execution time: $\max(T_{math}, T_{mem})$
- When $T_{math} = T_{mem}$, both utilization are maximized
- Decoding can add more tasks without introducing higher TPOT
- Note that the memory-bound is for one decoding task

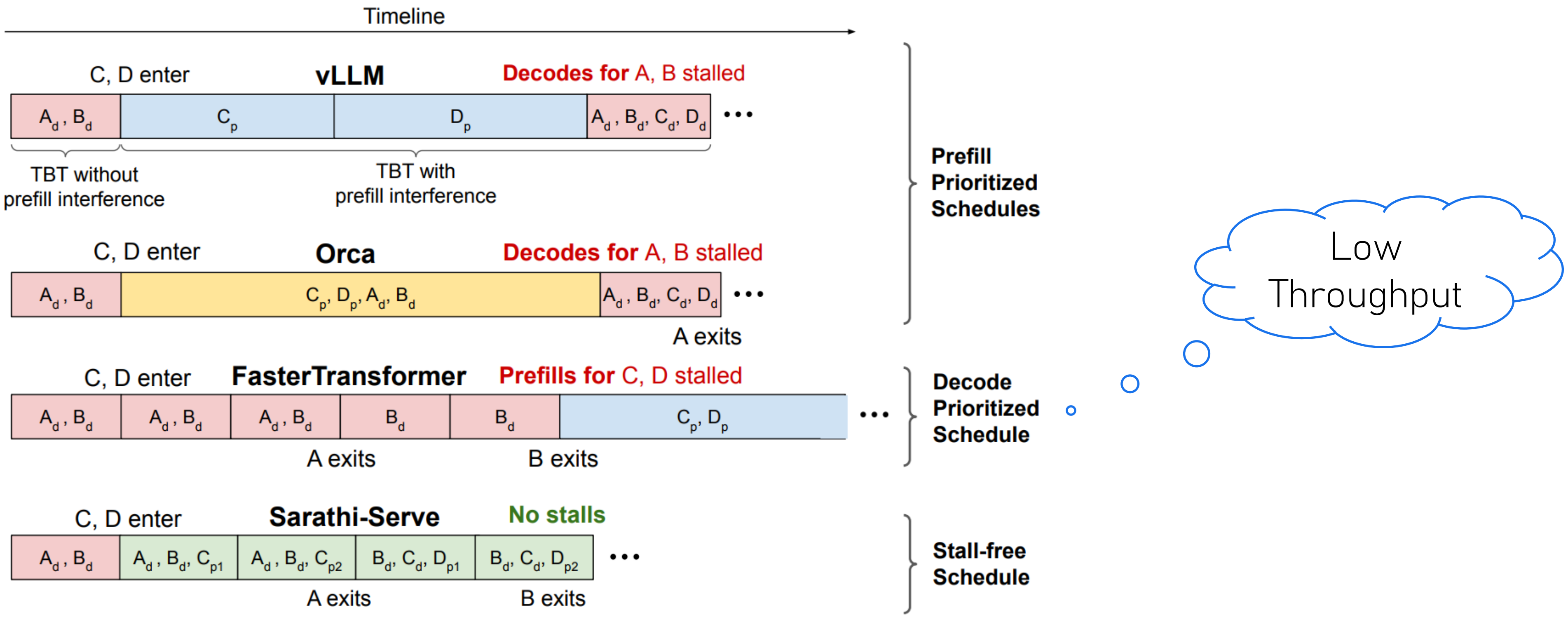
Background – Scheduling Policies



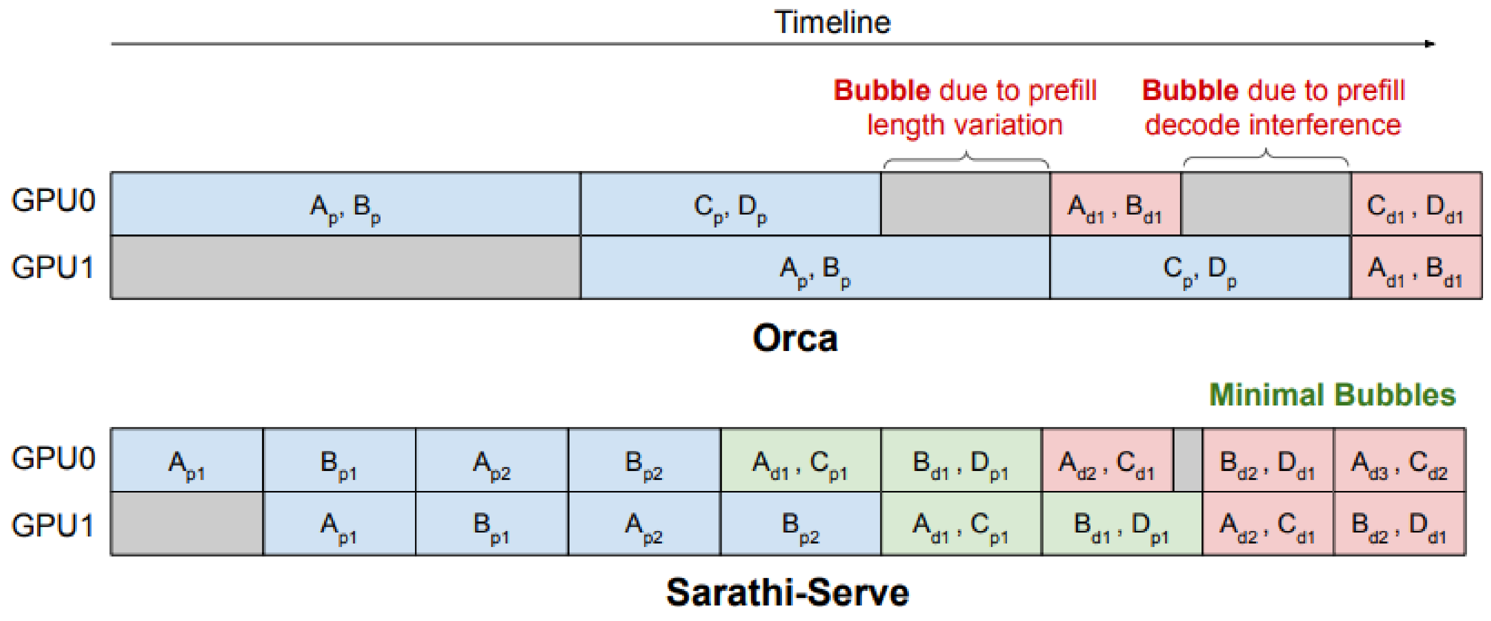
Background – Scheduling Policies



Background – Scheduling Policies



Background – Pipeline Parallelism Bubbles



Chunked Prefill

- Break the long prefill task into small slices
- Batch the prefill and decoding task together
- Saturate the computational resources without causing longer TPOT
- Higher throughputs without higher TPOT

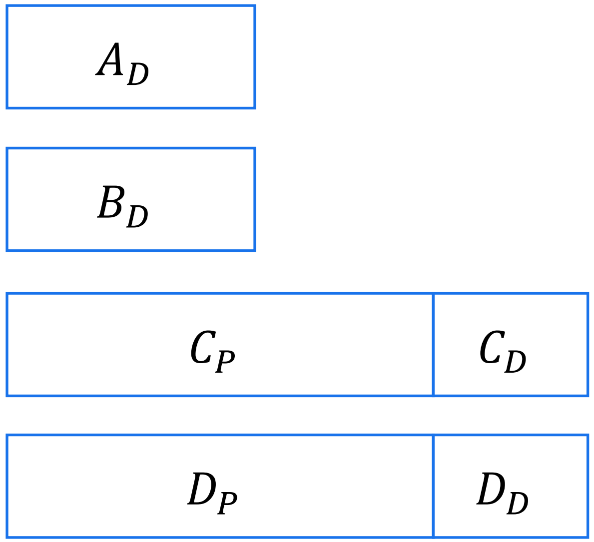
Stall-Free Batching

- Determine the budget of the maximum tokens in a batch under SLO
  - Larger budget -> fewer memory loads but longer latency
  - Smaller budget -> latency effect is smaller but it needs more memory operations
  - Tile-quantization: the prefill size should match the GPU tile size
  - For PP, different budgets cause different sizes of bubbles

Stall-Free Batching

- Determine the budget of the maximum tokens in a batch under SLO
  - Larger budget -> fewer memory loads but longer latency
  - Smaller budget -> latency effect is smaller but it needs more memory operations
  - Tile-quantization: the prefill size should match the GPU tile size
  - For PP, different budgets cause different sizes of bubbles
- Scheduling prefill and decoding based on the budget

Stall-Free Batching



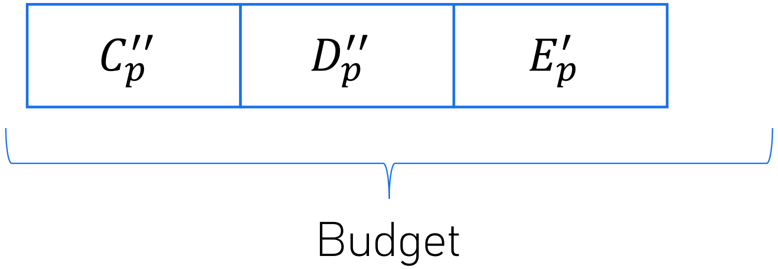
Stall-Free Batching

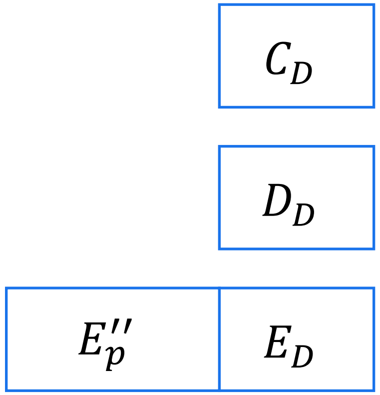
$$\underbrace{\boxed{A_D}\boxed{B_D}\boxed{C'_p}\boxed{D'_p}}_{\text{Budget}}$$

C''_p	C_D

D''_p	D_D

Stall-Free Batching





Evaluation – Settings

- Models: Mistral-7B, Yi-34B, LLaMA2- 70B, Falcon-180B
- A100 80GB, A40 48GB
- Yi-34B: TP=2, LLaMA2- 70B, Falcon-180B: TP4-PP2

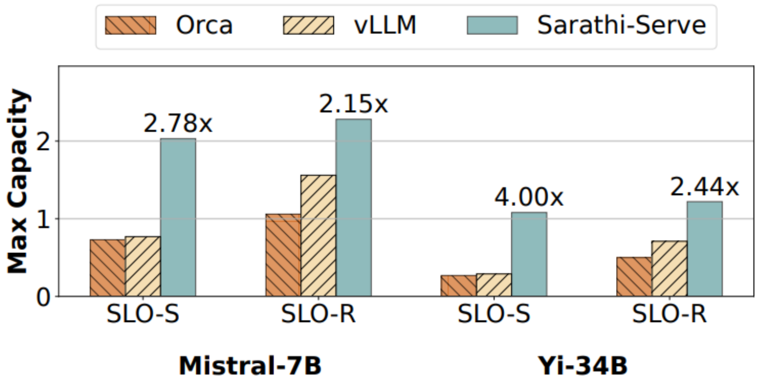
Evaluation – Workloads & Metrics

- Workloads: openchat_sharegpt4, arxiv_summarization
- Time: Poisson distribution
- Metrics: median TTFT and P99 TPOT

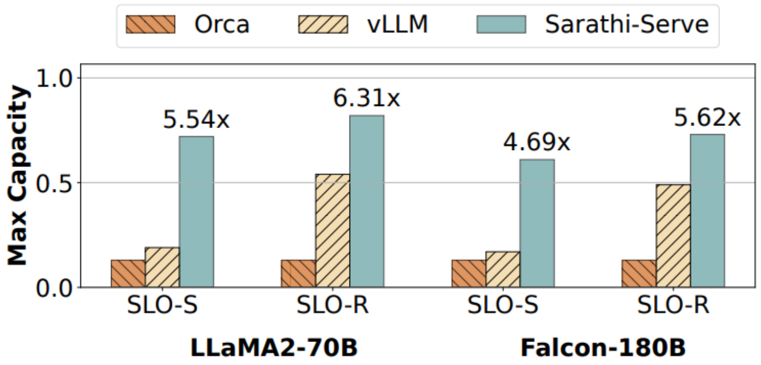
Model	***relaxed* SLO P99 TBT (s)**	***strict* SLO P99 TBT (s)**
Mistral-7B	0.5	0.1
Yi-34B	1	0.2
LLaMA2-70B	5	1
Falcon-180B	5	1

Table 3: SLOs for different model configurations.

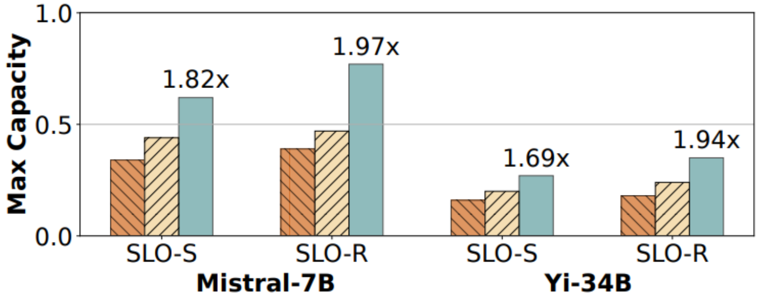
Evaluation – Capacity



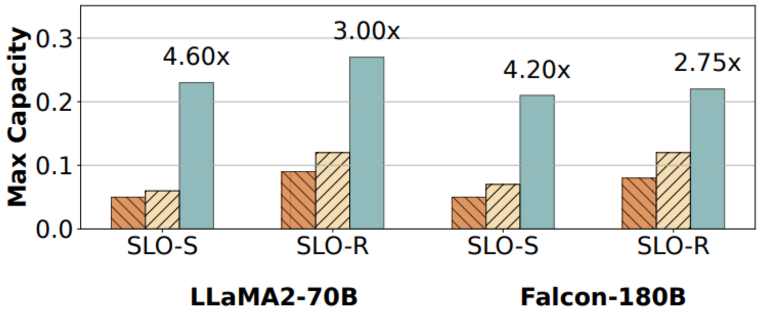
(a) Dataset: *openchat_sharegpt4*.



(a) Dataset: *openchat_sharegpt4*.

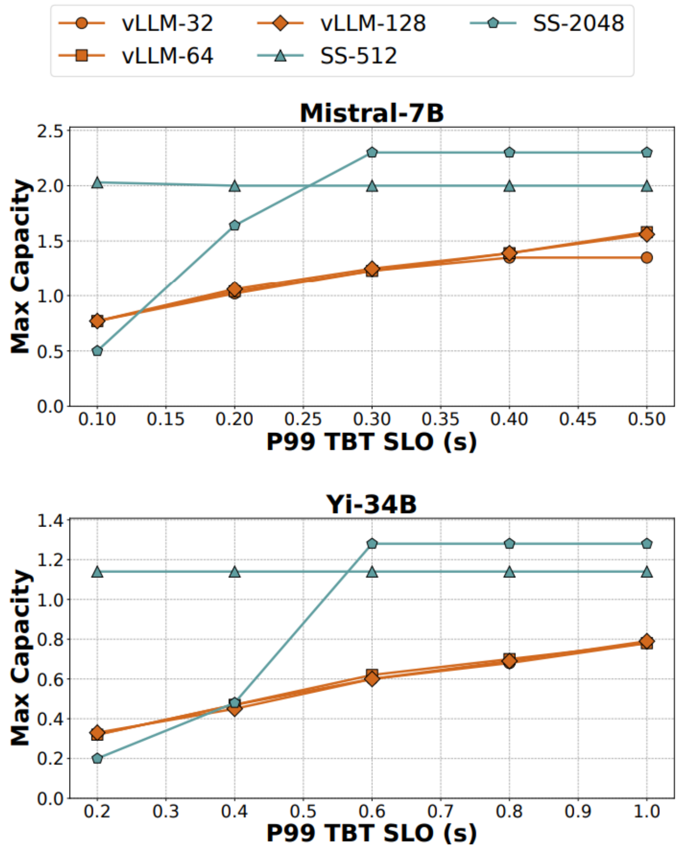


(b) Dataset: *arxiv_summarization*.

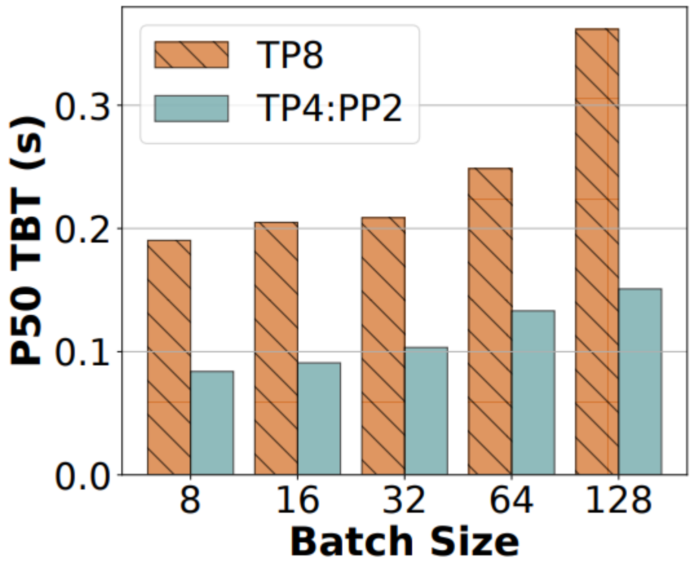


(b) Dataset: *arxiv_summarization*.

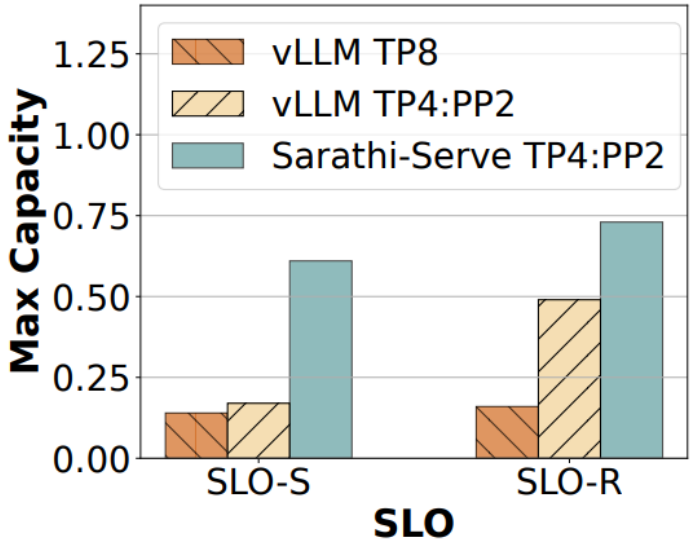
Evaluation – Throughput & Latency



Evaluation - PP

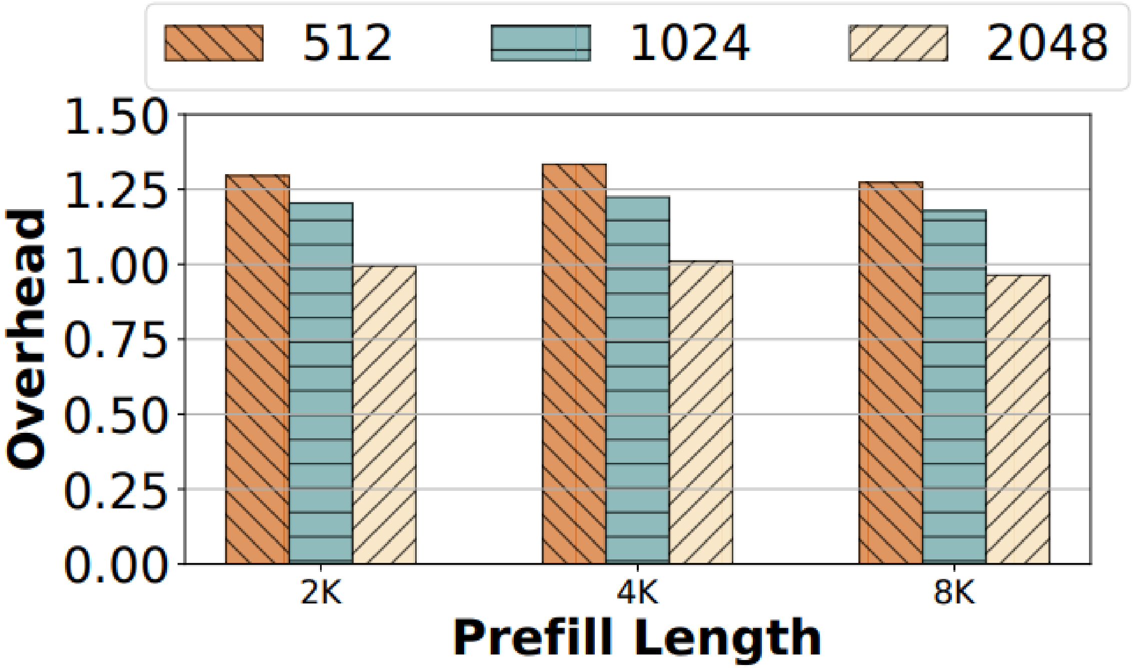


(a) TBT (Falcon-180B).



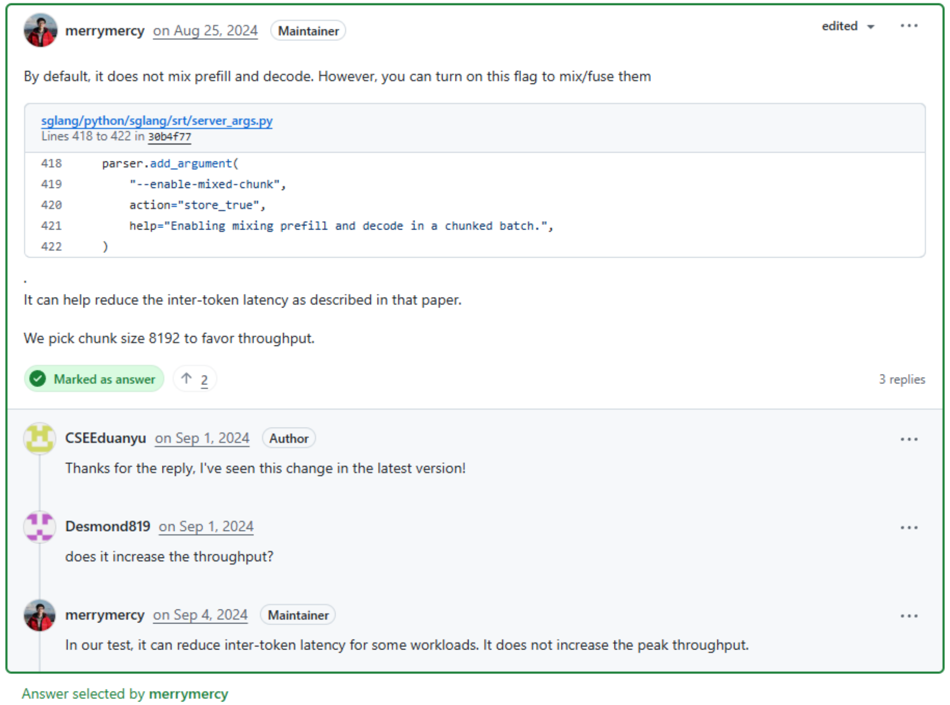
(b) Capacity (Falcon-180B).

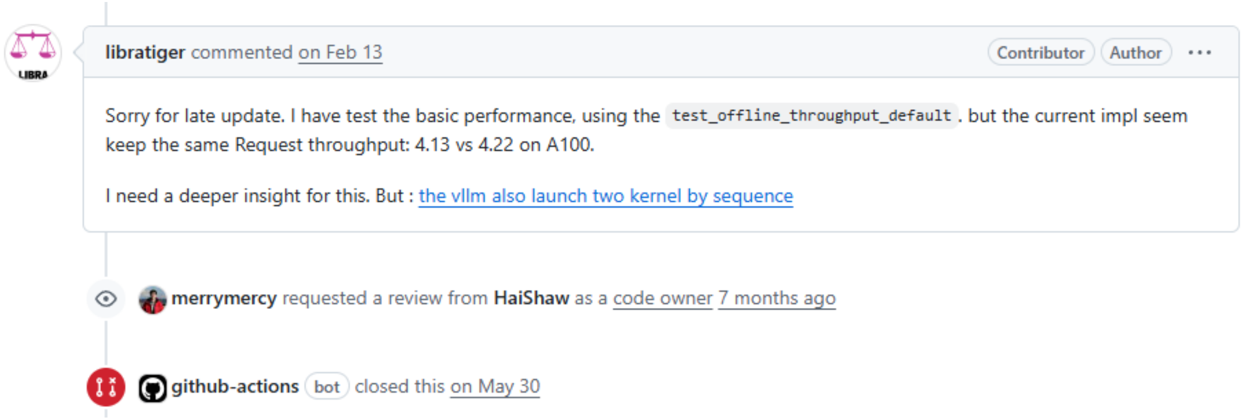
Evaluation – Ablation



Scheduler	***openchat_sharegpt4***		***arxiv_summarization***	
	P50 TTFT	P99 TBT	P50 TTFT	P99 TBT
hybrid-batching-only	0.53	0.68	3.78	1.38
chunked-prefills-only	1.04	0.17	5.38	0.20
Sarathi-Serve (combined)	0.76	0.14	3.90	0.17

Test On SGLang





Homework

- Read the paper **CacheBlend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion**, summarize the paper, specifically, including the points below:
  - What are the motivations/challenges of this work?
  - How does the design of this paper address the challenges?
  - How does the paper evaluate its design (experiment settings, workloads, metrics)?
  - How does the evaluation prove its claims?
- Note that it is essential to logically connect the motivation, design, and evaluation, rather than merely listing some points.
- Related link:
  - Paper of CacheBlend: https://arxiv.org/pdf/2405.16444

Q & A