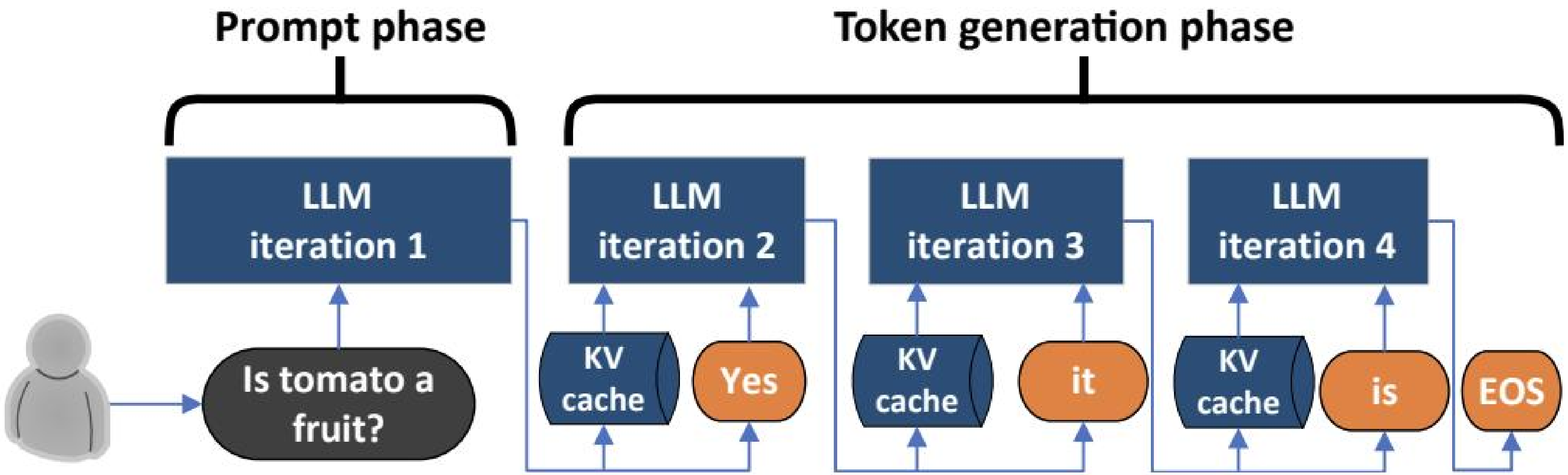
Prefill-Decode Disaggregation

Chenye Wang

Dec 11, 2024

[1] Splitwise: Efficient Generative LLM Inference Using Phase Splitting, 2023.
[2] DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving, OSDI24.
[3] MuxServe: Flexible Spatial-Temporal Multiplexing for Multiple LLM Serving, ICML, 2024.

LLM inference process



Prefill:

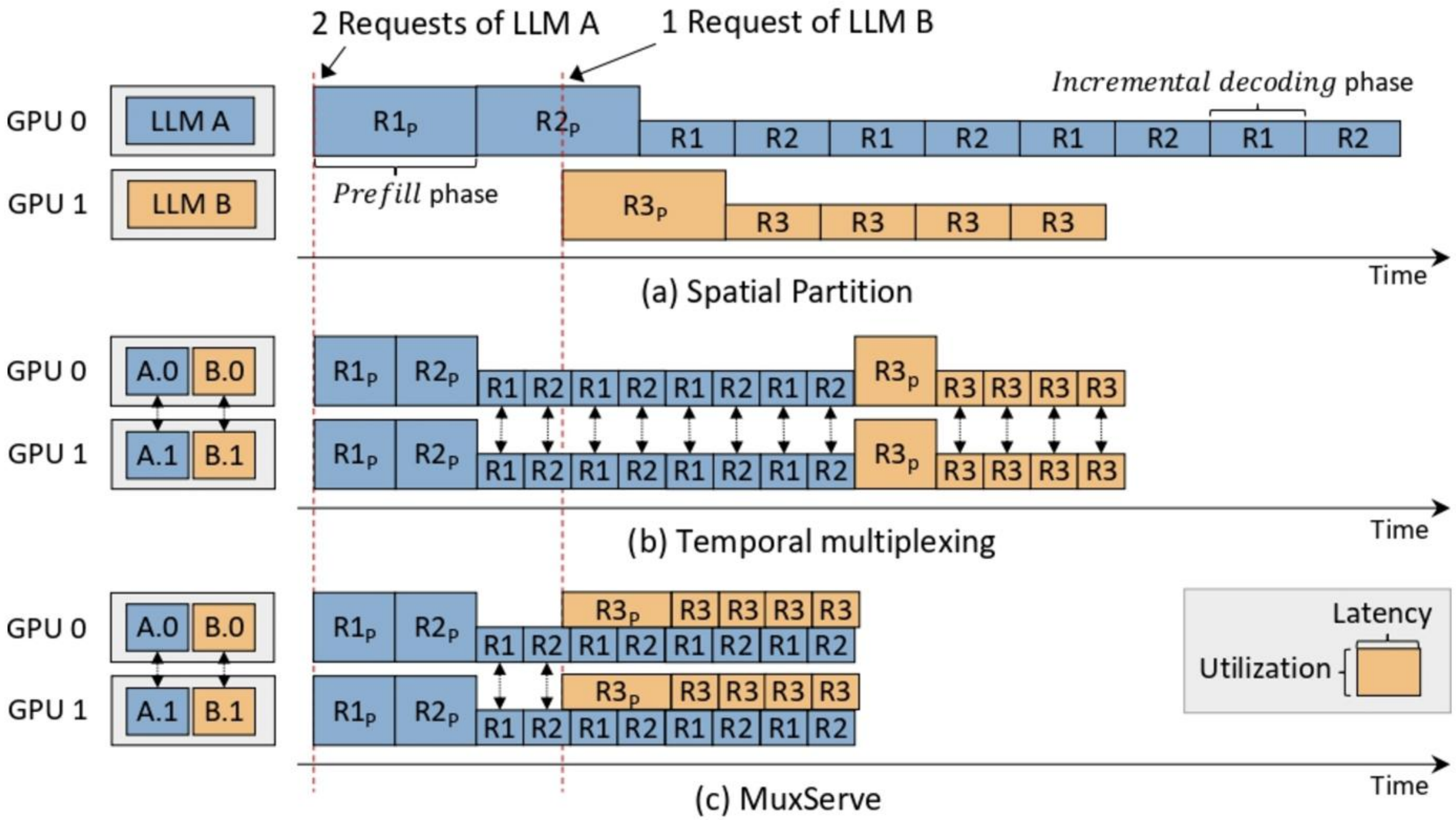
compute intensive

Decode:

memory intensive

majority of E2E latency

Spatial-Temporal Multiplexing



Service demand

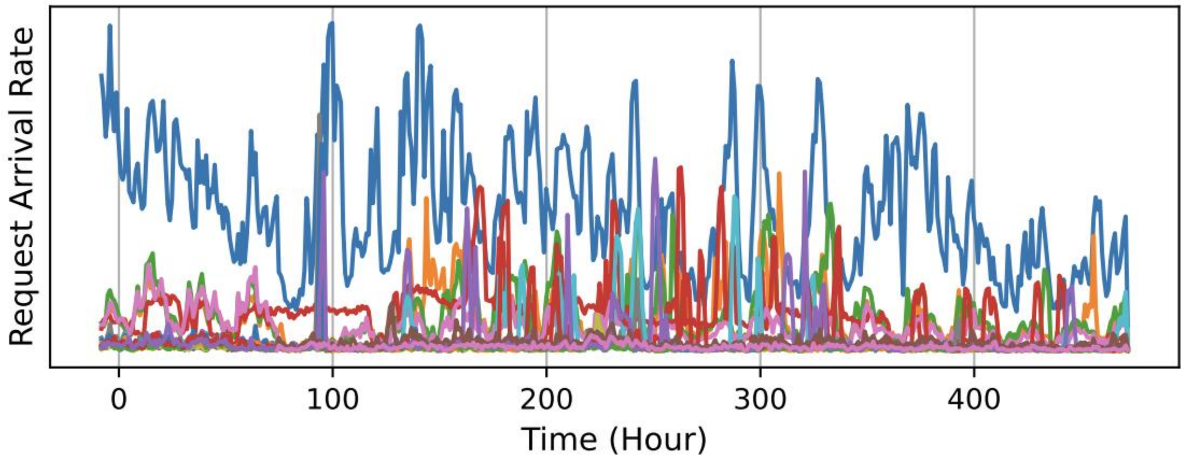
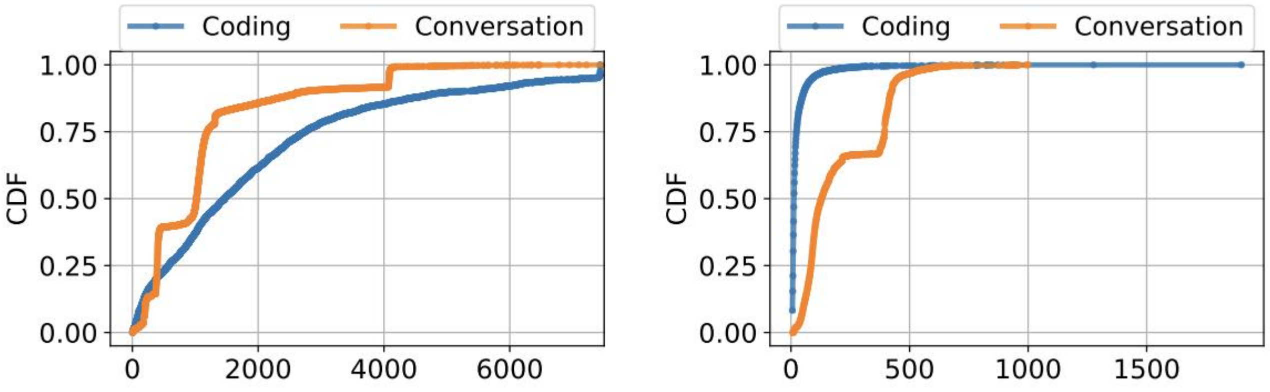


Figure 2. The dynamic request arrival rates of different LLMs over a 20 day period.



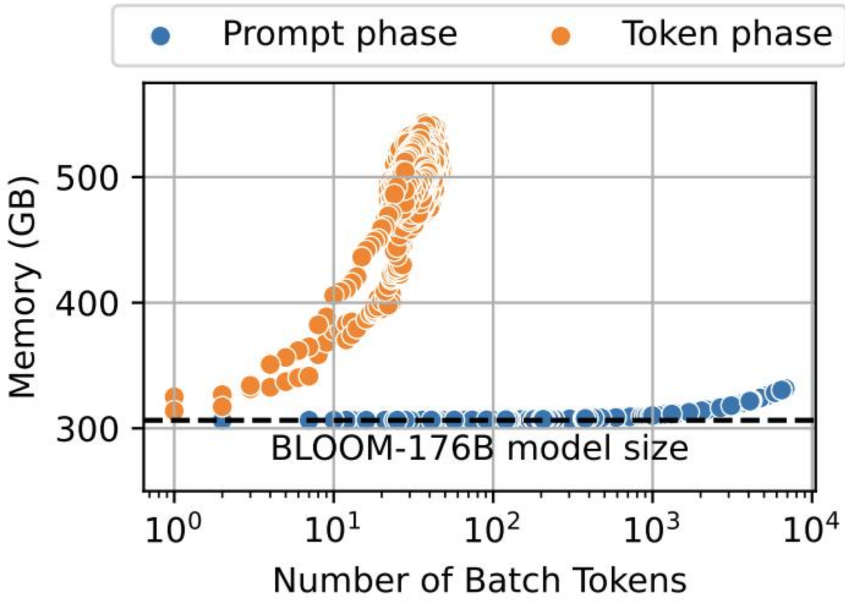
(a) Prompt input tokens. (b) Generated output tokens.

Fig. 3: Distribution for prompt and generated tokens.

- Insight 1:
  - LLM popularity varies significantly.
  - Different inference services may have widely different prompt and token distributions.

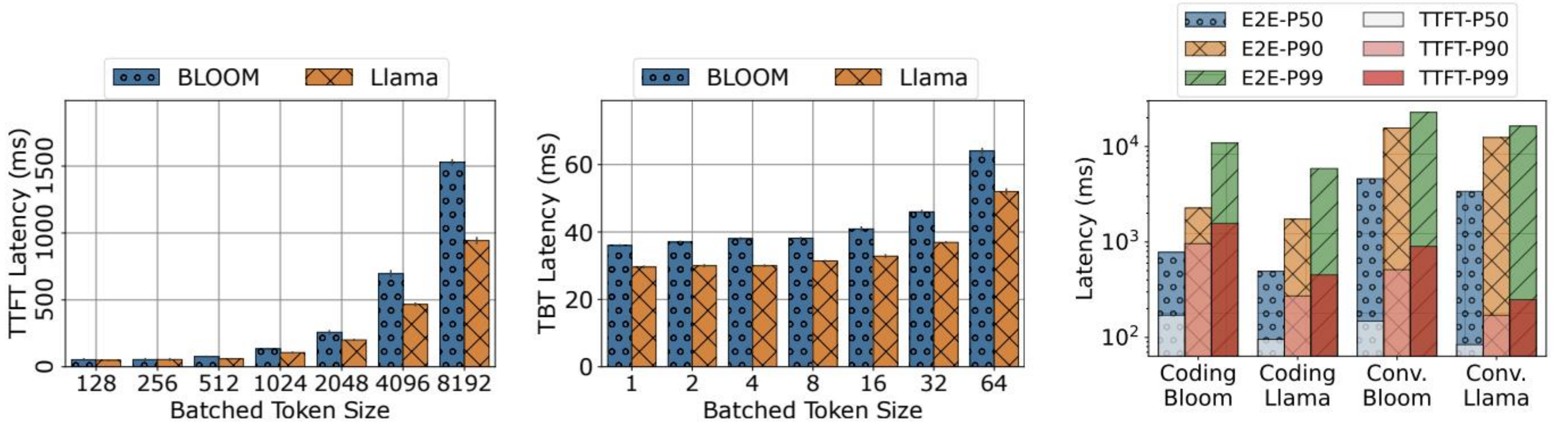
Memory footprint

- Insight 2:
  - The decode phase is memory intensive



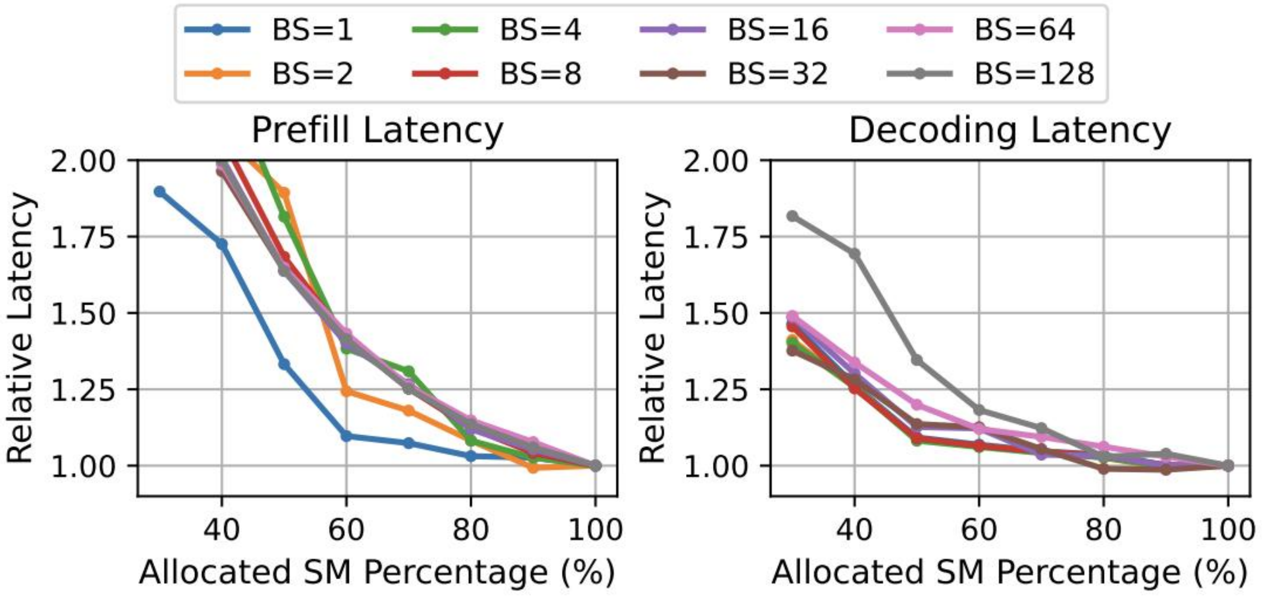
Model	Parameters (GB)	#Layers	#Heads	Head Size	KV Cache of 1k Tokens (MB)
LLaMA-7B	14	32	32	128	262.1
LLaMA-13B	26	40	40	128	409.6
LLaMA-30B	60	60	52	128	798.72
LLaMA-65B	130	80	64	128	1310.7

Latency



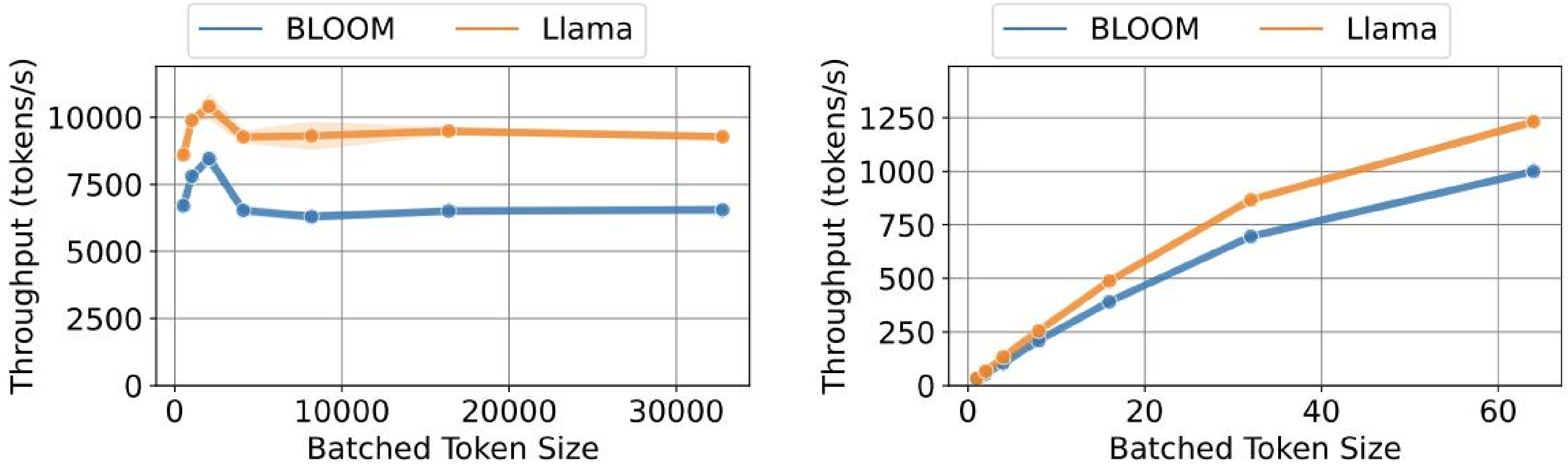
(a) TTFT by prompt size. (b) TBT by batch size. (c) Latencies on prod traces (no batching).

- Insight 3:
  - The prefill phase is compute intensive.
  - The majority of E2E time is spent in the decode phase.



Throughput

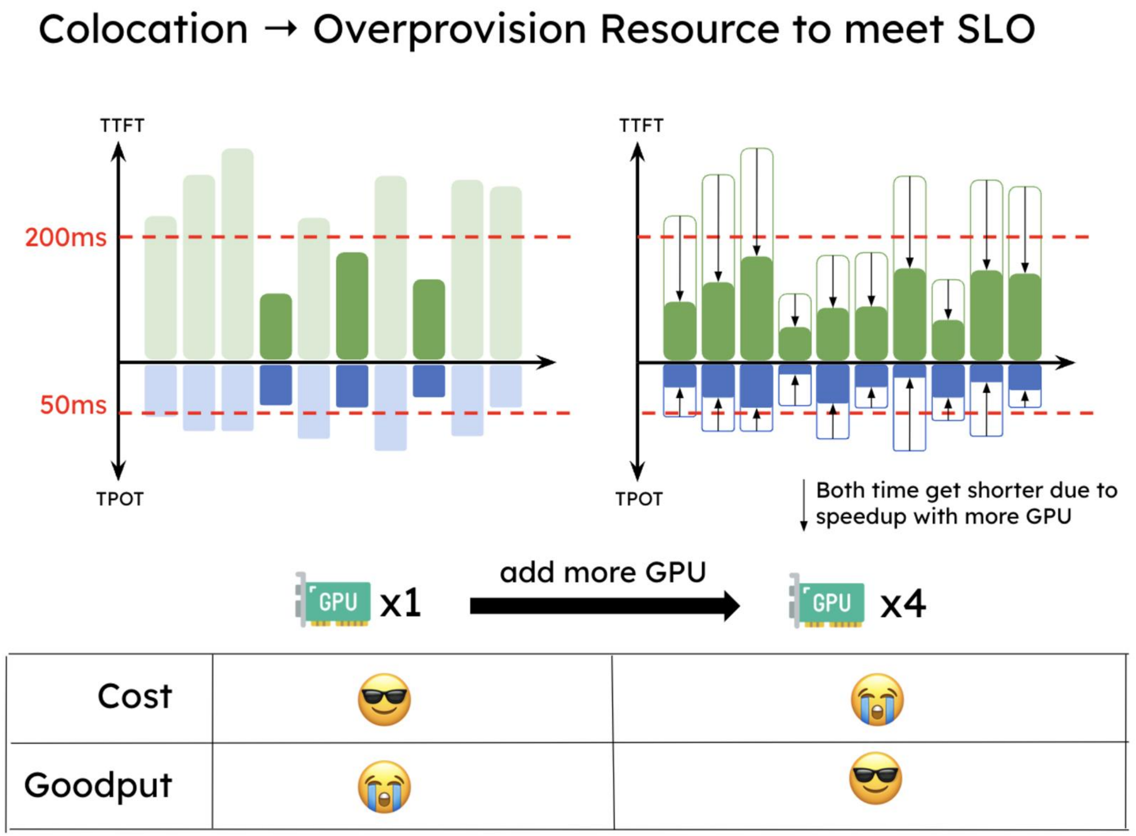
- Insight 4:
  - The prompt phase batch size should be limited to ensure good performance.
  - In contrast, batching the token generation phase yields high throughput without any downside.

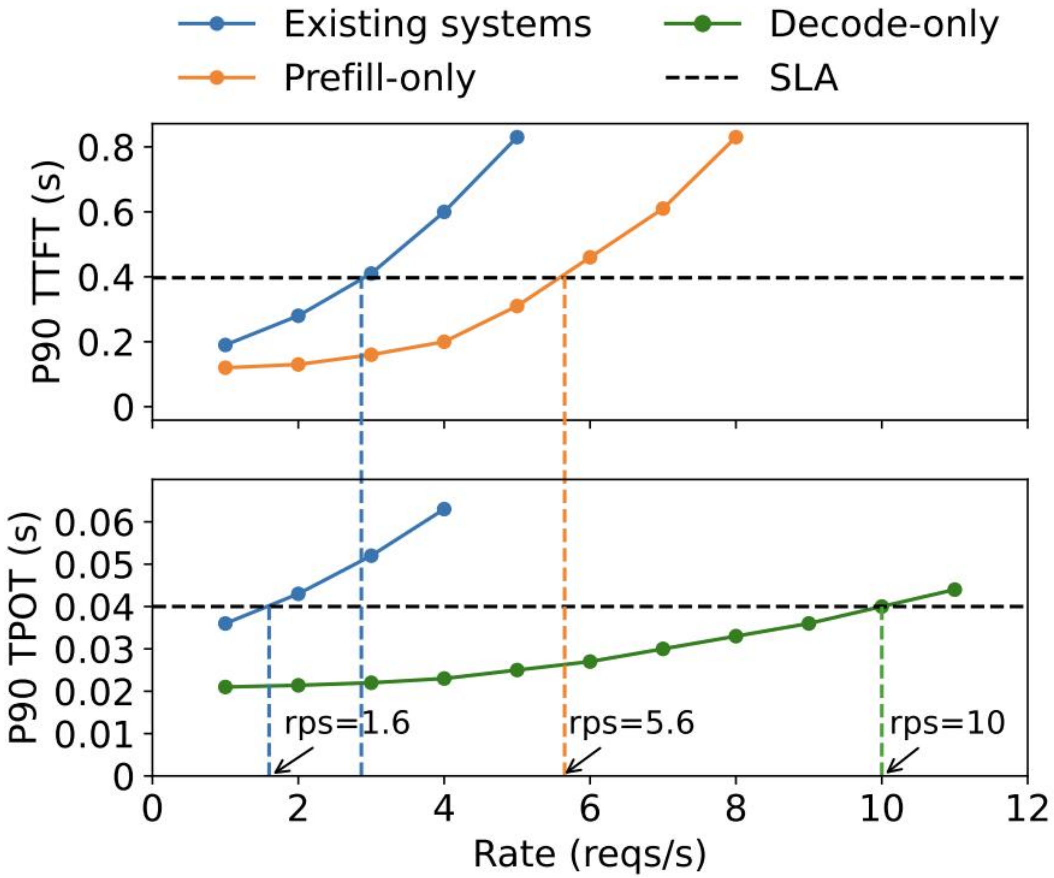


(a) Prompt phase.   (b) Token generation phase.

Goodput (SLO)

- Collocating prefill and decode causes Interference.
- By allocating 2 GPUs for prefill and 1 GPU for decoding, we can get 2x goodput.





GPU hardware selection

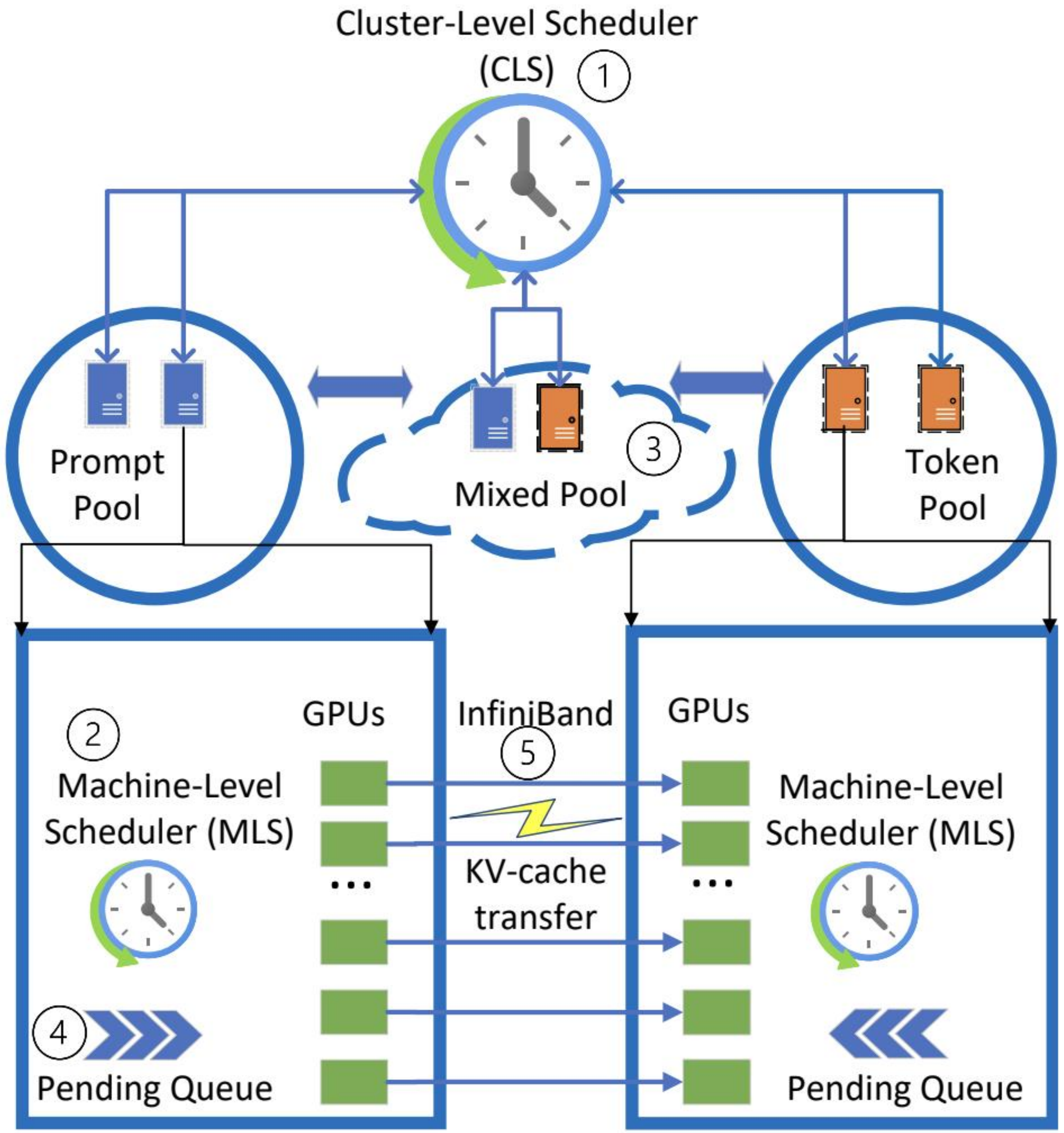
- Decode phase can be run on less compute_x0002_capable hardware to reduce device cost.

	Coding			Conversation		
	A100	**H100**	**Ratio**	**A100**	**H100**	**Ratio**
TTFT	185 ms	95 ms	0.51×	155 ms	84 ms	0.54×
TBT	52 ms	31 ms	0.70×	40 ms	28 ms	0.70×
E2E	856 ms	493 ms	0.58×	4957 ms	3387 ms	0.68×
Cost [5]	$0.42	$0.52	1.24×	$2.4	$3.6	1.5×
Energy	1.37 Whr	1.37 Whr	1×	7.9 Whr	9.4 Whr	1.2×

TABLE IV: P50 request metrics on A100 vs. H100 without batching on Llama-70B.

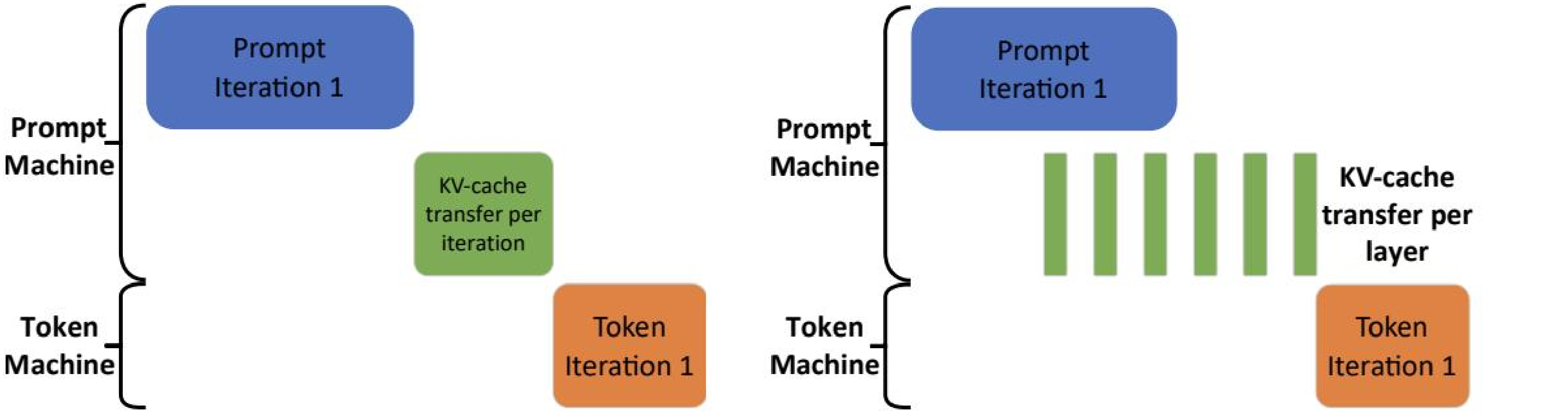
Splitwise

- Machine pool management
  - Prompt pool
  - Token pool
  - Mixed pool
    - Dynamic adjustment
- Scheduler
  - Cluster-level scheduling
    - Request routing
    - Join the Shortest Queue (JSQ)
  - Machine-level scheduling
    - FCFS
    - Batching



Layer-wise KV-cache transfer

- After each layer in the LLM is calculated in the prompt machine, the KV cache corresponding to that layer is immediately transferred to the token machine.



(a) Serialized KV-cache transfer. (b) Optimized KV-cache transfer per-layer during prompt phase.

TODO

- DistServe
  - High Node-Affinity Placement Algorithm
  - Low Node-Affinity Placement Algorithm
- MuxServe
  - Enumeration-based Greedy Placement Algorithm
  - Adaptive Batch Scheduling Algorithm