Sarathi: Efficient LLM Inference with Low Tail Latency

Amey Agrawal[1], Nitin Kedia[2], Ashish Panwar[2], Jayashree Mohan[2],
Nipun Kwatra[2], Bhargav Gulavani[2], Alexey Tumanov[1], Ramchandran Ramjee[2]
[1]Systems for AI Lab, Georgia Tech, [2]Microsoft Research

SAIL

How to achieve high throughput with tail latency in LLM Inference workloads? 🤔

Observation 1: The decode time per token is order(s) of magnitude higher than prefill, especially at small batch sizes.

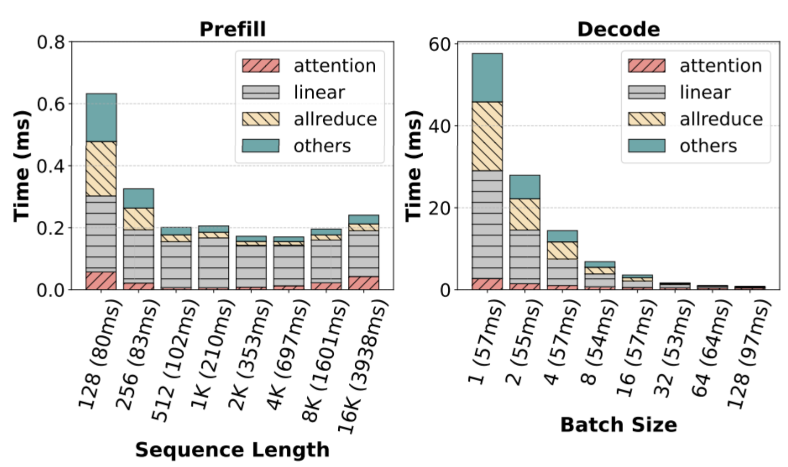
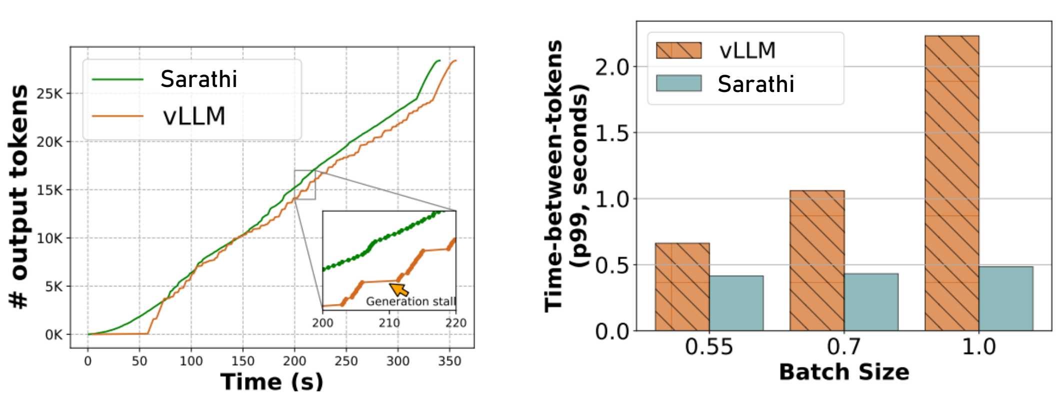


Figure 1: Per-token prefill and decode time.

Observation 2: Interleaved prefill iterations increases TBT tail latency, which gets exacerbated as batch size increases.



(a) Generation stall.
(b) High tail latency.

Figure 2: (a) A generation stall lasting over several seconds - a common phenomenon in vLLM. (b) State-of-the-art serving systems such as vLLM are susceptible to high tail latency.

Idea: Split large prefills into smaller chunks to generate uniform sized compute units 💭

Chunked Prefills: Split a large prefill request into multiple uniform sized chunks.

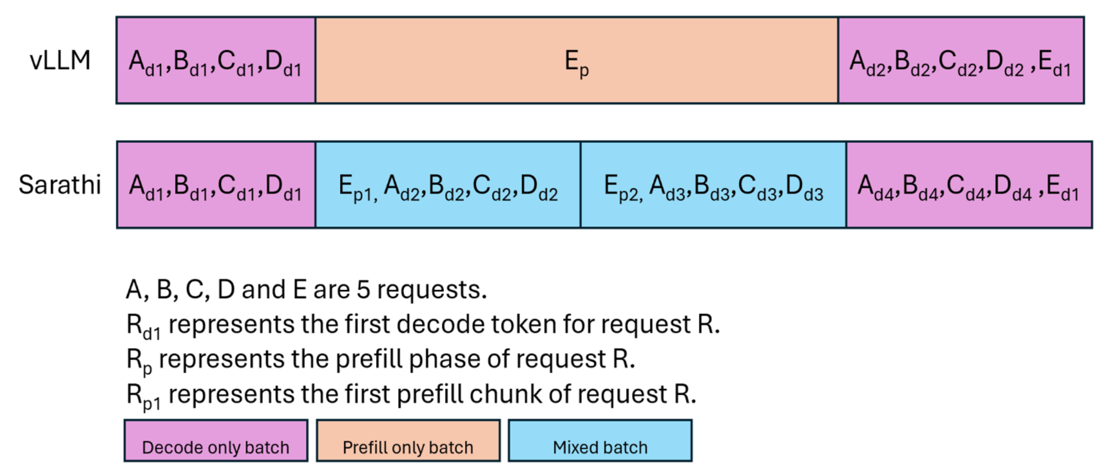


Figure 3: Sarathi scheduling with chunked prefills.

Decode-maximal Batching: GPU is mostly idle during decoding – waiting for model weights to arrive from memory. Piggyback decode tokens with prefills to improve GPU utilization

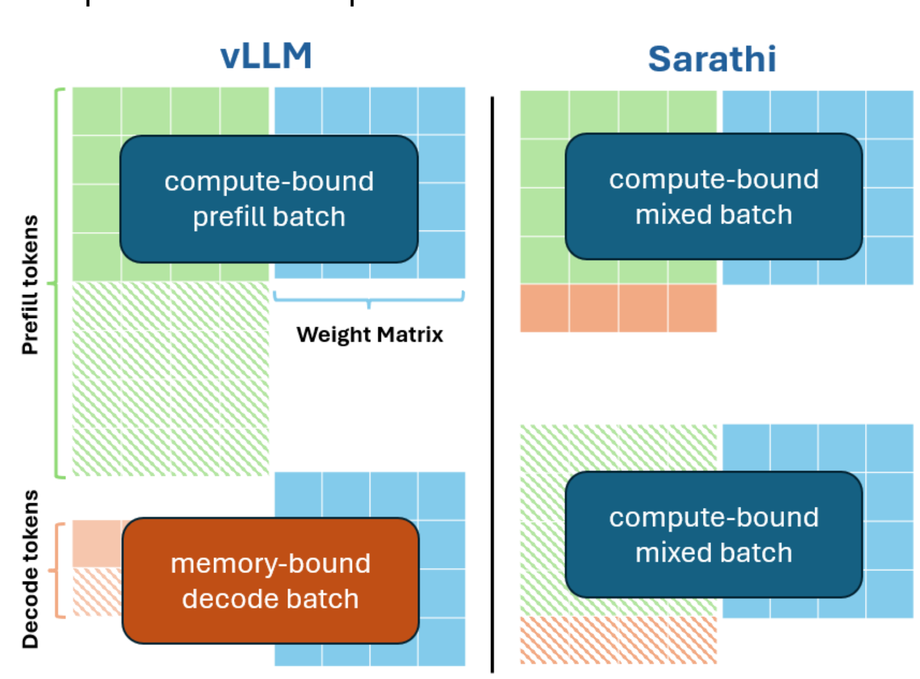


Figure 4: Saturating compute utilization with mixed batching.

3-10x higher serving capacity compared to vLLM under Under latency constraints 🎉

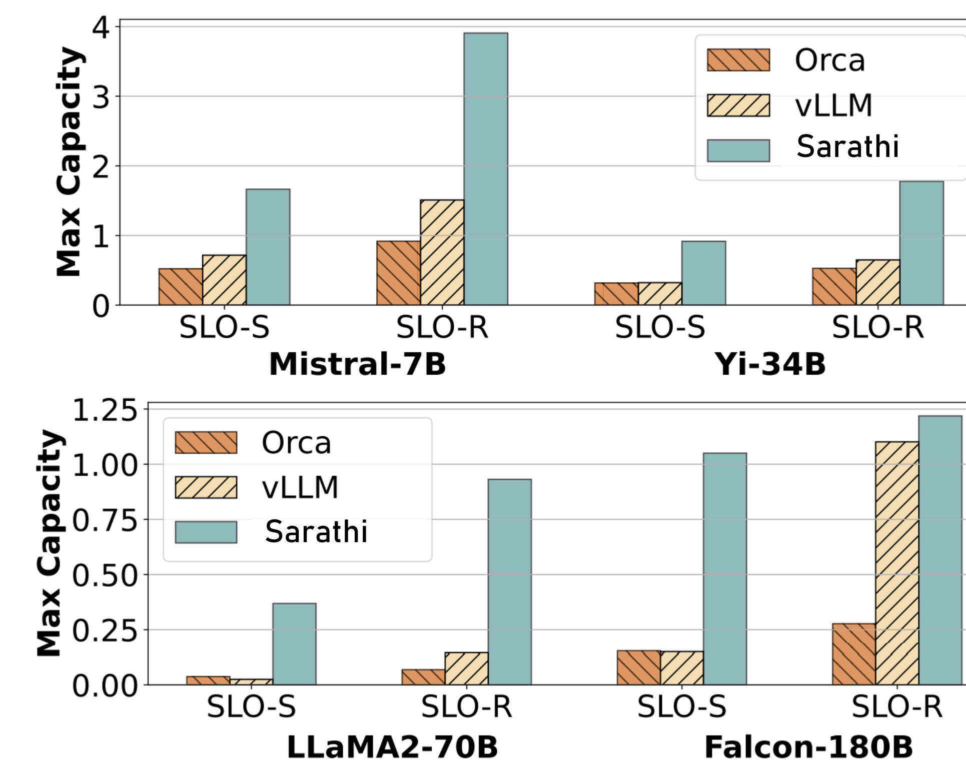
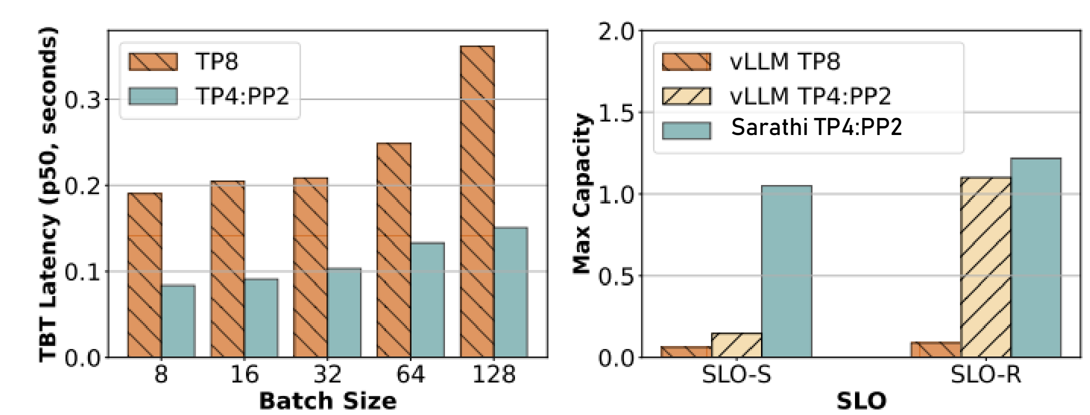


Figure 5: Capacity with different schedulers under strict (SLO-S) and relaxed (SLO-R) latency SLOs on Openchat-ShareGPT4 trace.



(a) TBT (Falcon-180B). (b) Capacity (Falcon-180B).

Figure 6: Efficiently scaling inference across nodes using bubble-free pipeline parallelism enabled by chunked prefills.

Scan the QR code to learn more

Correspondence at:
ameyagrawal@gatech.edu