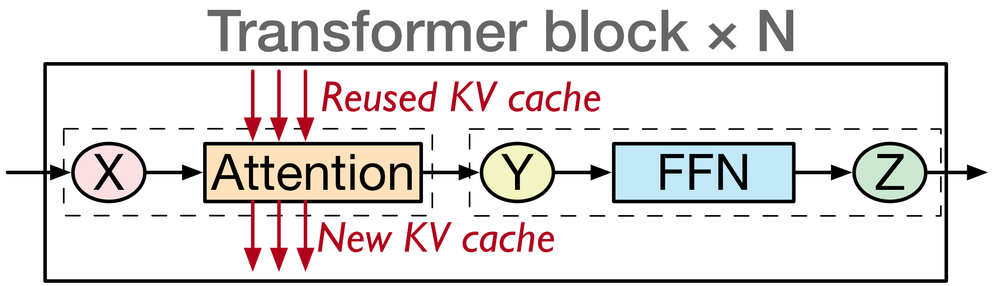
Optimizing SLO-oriented LLM Serving with PD-Multiplexing

Presenter: Yang Zhou

Oct 30, 2025

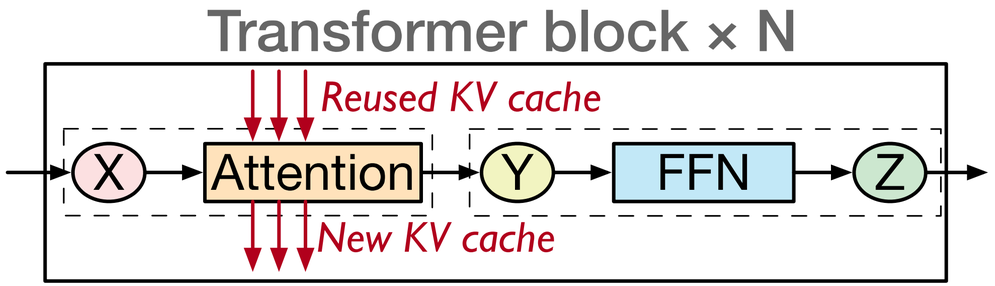
Background: LLM inference

✦LLM architecture

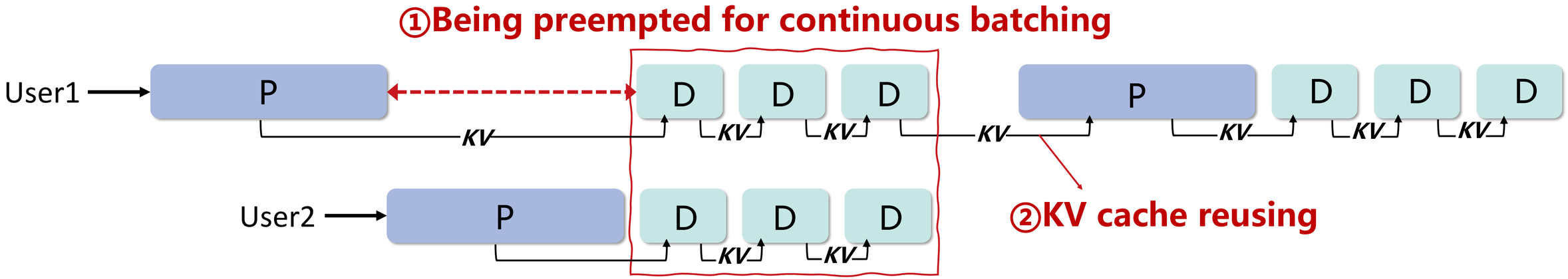


Background: LLM inference

✦LLM architecture

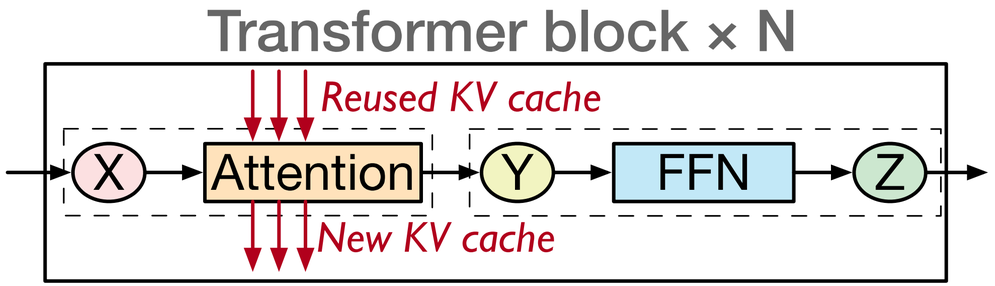


✦LLM inference process

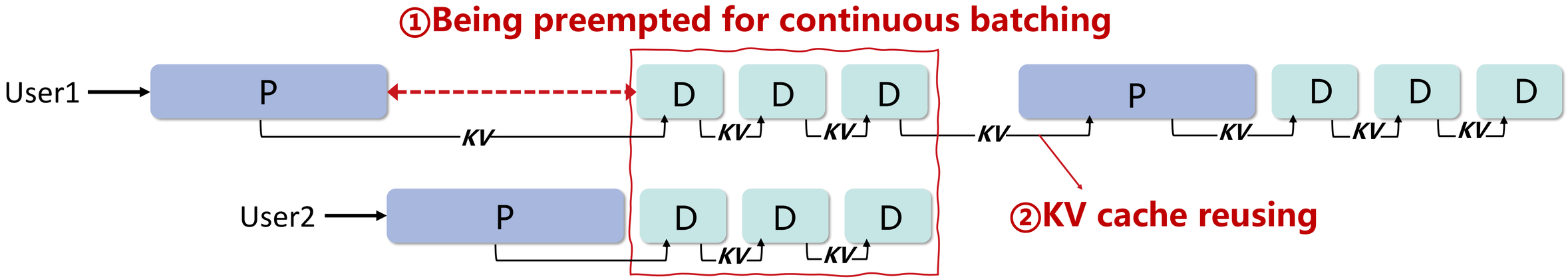


Background: LLM inference

✦LLM architecture



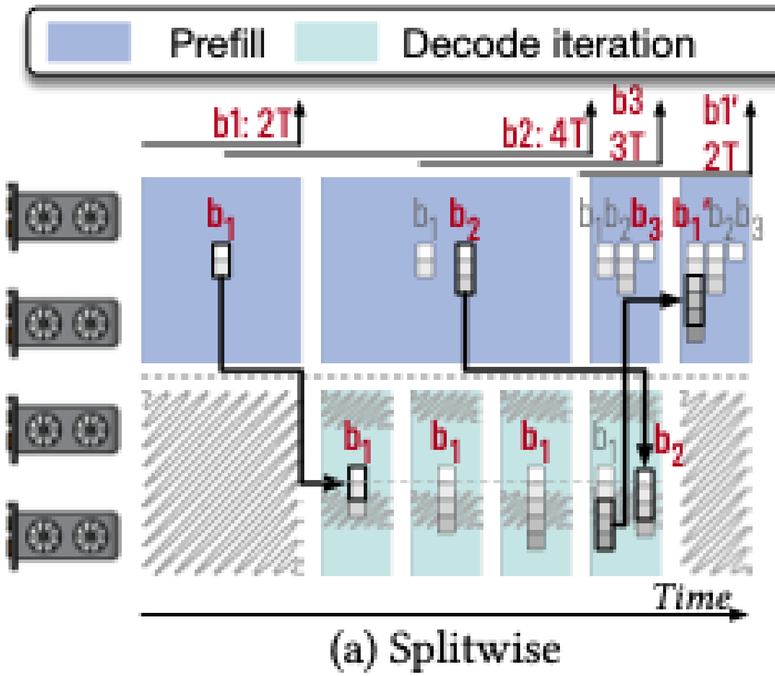
✦LLM inference process



How to guarantee SLO, while improving goodput?

Existing solutions

✦Static PDD, Dynamic PDD, Chunked-prefill

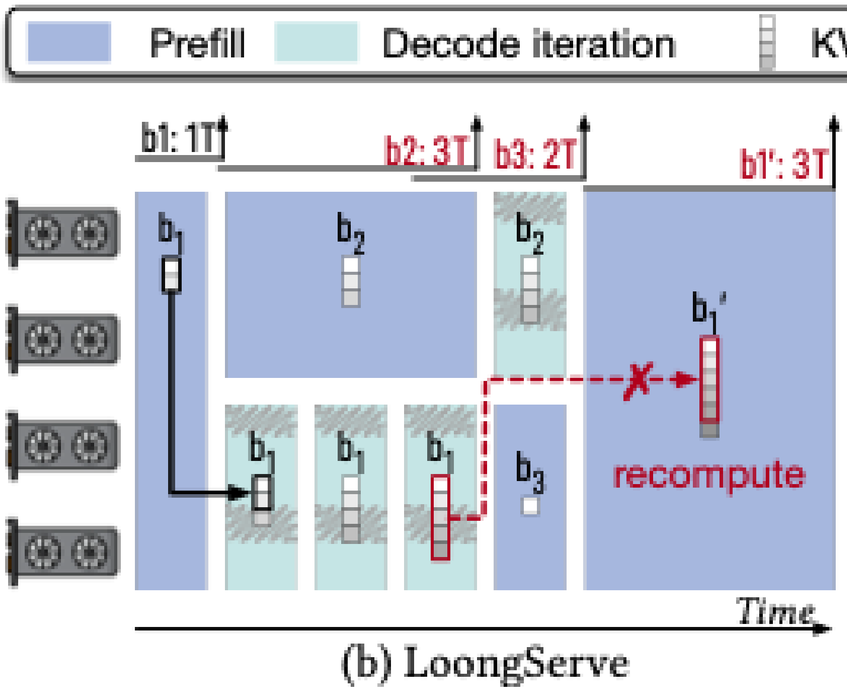


(a) Splitwise

- ✓ Meets SLO requirements for TTFT and TBT
- ✗ Cannot adapt to dynamic workload changes, leading to resource waste on certain nodes
- ✗ KV cache and model weights are separated, shrinking the KV cache pool, lowering cache hit rate, and degrading multi-turn dialogue performance
- ✗ Relies on high-performance interconnects between nodes to transfer KV data, requiring complex point-to-point communication implementations

Existing solutions

✦Static PDD, Dynamic PDD, Chunked-prefill

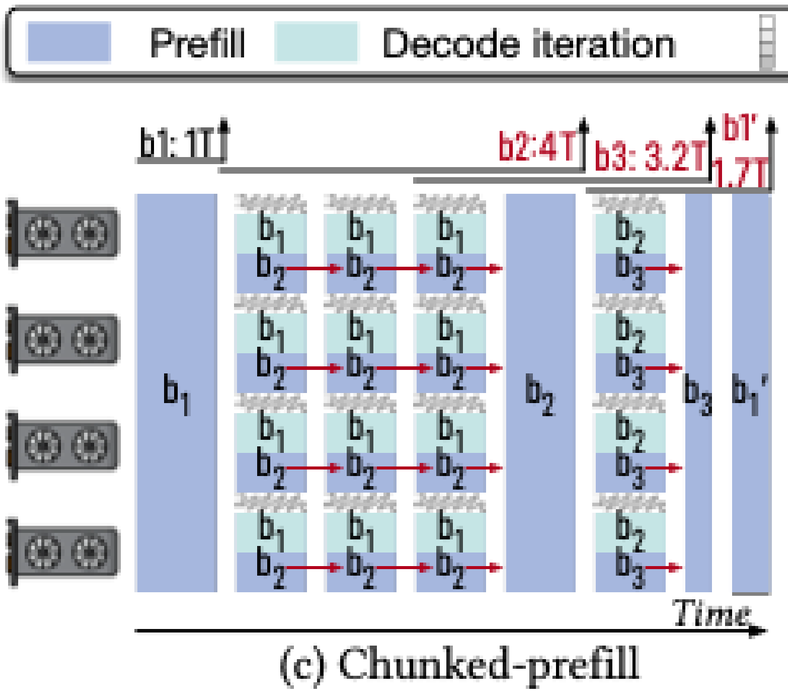


(b) LoongServe

- ✓ Meets SLO requirements for TTFT and TBT
- ✓ Can dynamically adjust the resource ratio between Prefill and Decode to adapt to workload changes
- ✗ Resource allocation changes are at the GPU granularity — if either compute or memory needs adjustment, both must change — thus resource waste still exists
- ✗ Requires KV cache migration and cannot reuse KV cache across requests, resulting in redundant recomputation

Existing solutions

✦Static PDD, Dynamic PDD, **Chunked-prefill**



(c) Chunked-prefill

- ✓ Meets SLO requirements for TBT
- ✓ Can dynamically adjust the resource ratio between Prefill and Decode to adapt to workload changes
- ✗ Requires balancing between SLO and high utilization — a small TBT SLO target can lead to idle resources
- ✗ For long sequences, as the reused KV cache computation increases, the system may even fail to meet the TBT SLO

Drift: LLM Inference with Efficient SLO Guarantee

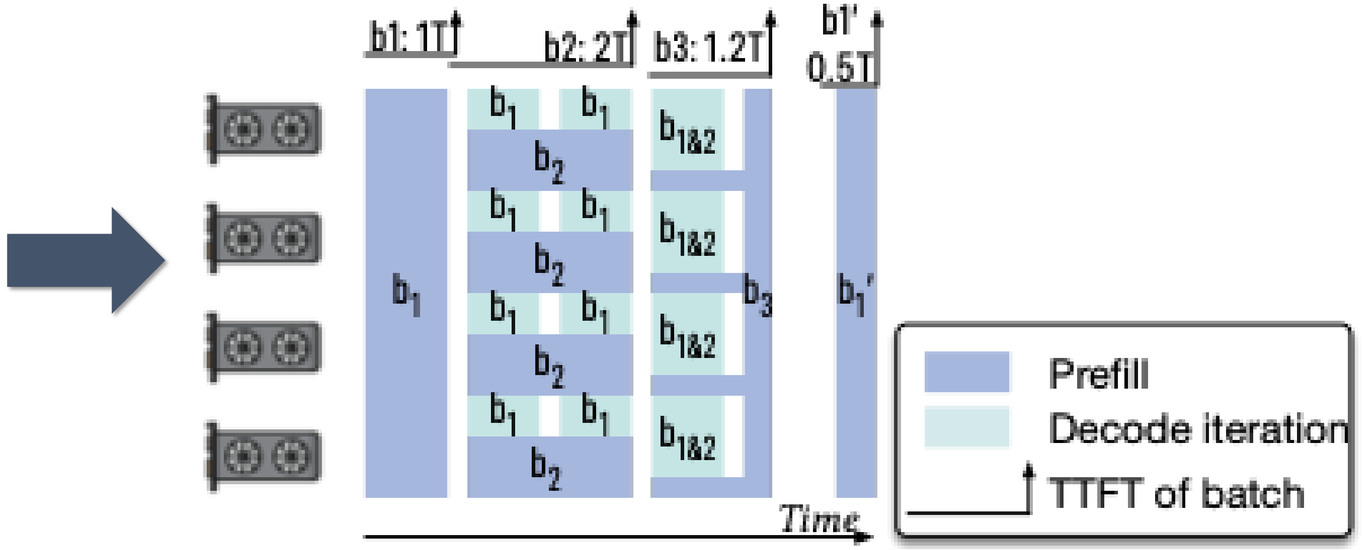
✦Design goals

- **✦Freely match the compute ratio between Prefill and Decode (PD)**
  - ✦Allow Prefill/Decode resources to change dynamically with the workload
- **✦Decouple compute allocation from memory**
  - ✦Changes in compute resources do not affect memory layout—especially the design of the KV cache pool
- **✦PD execution is mutually independent**
  - ✦Prefill and Decode only perform necessary synchronization, with no need to trade off between SLO and utilization

A New Paradigm for LLM inference: Spatial Prefill–Decode Co-Execution

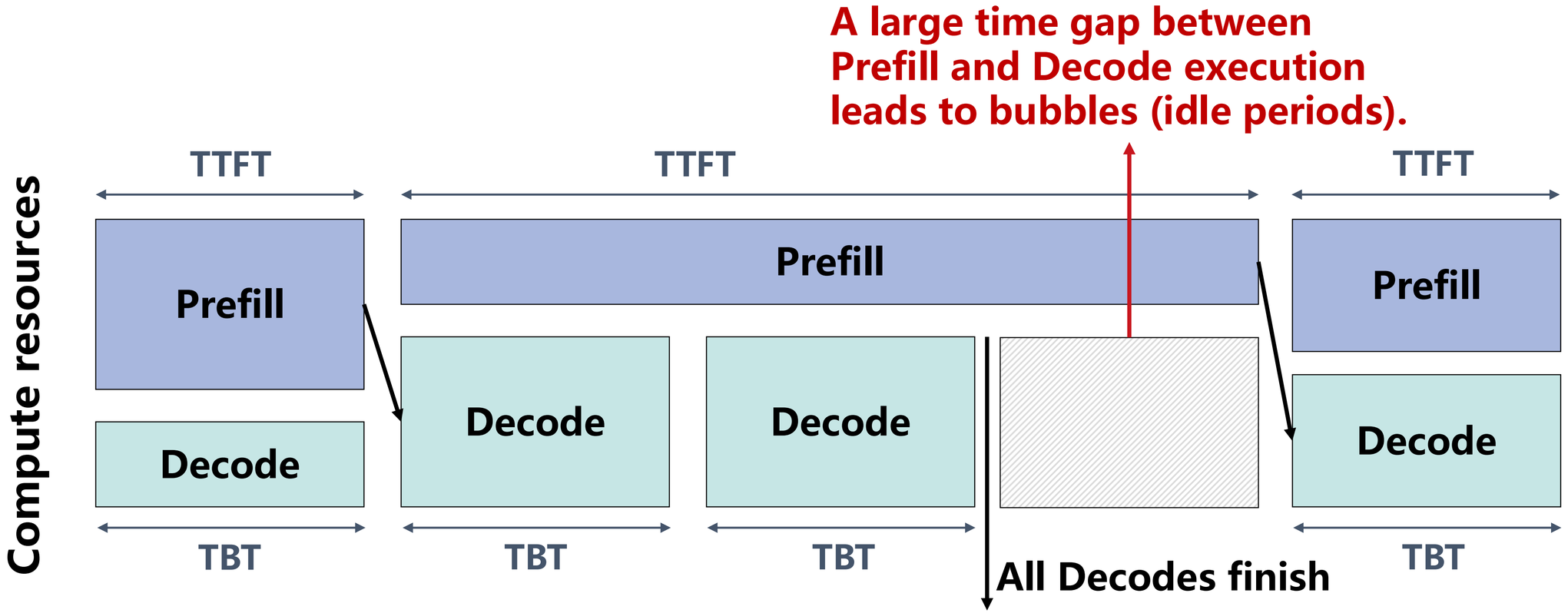
✦**Using spatial multiplexing, Prefill and Decode are co-located across all GPUs within a single node. By adjusting the number of SMs allocated to each, the system can meet the desired SLO requirements.**

- ✓ Flexible PD compute ratio matching
- ✓ Decoupled compute allocation from memory
- ✓ Independent execution of Prefill and Decode



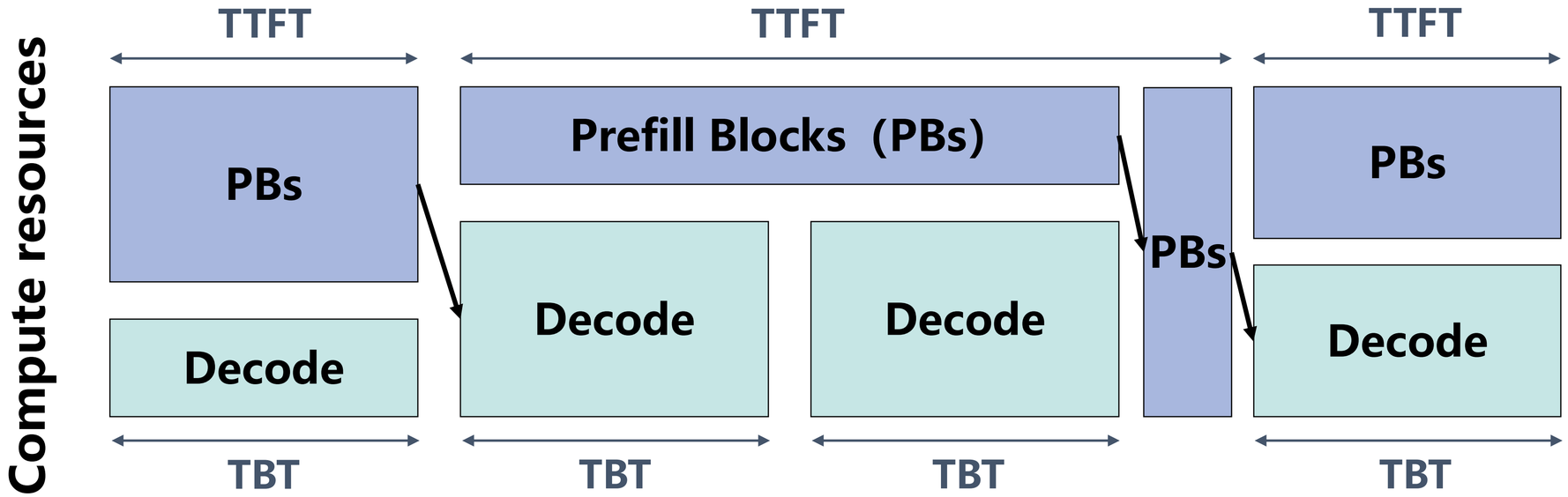
Prefill–Decode Spatial Multiplexing Mechanism

✦Basic Prefill–Decode Spatial Multiplexing Execution



Prefill–Decode Spatial Multiplexing Mechanism

✦Split Prefill by Transformer Blocks to align Prefill and Decode execution times, thereby reducing idle bubbles.



Prefill–Decode Spatial Multiplexing Mechanism

✦SLO-Aware Scheduling Method

- **Prioritize meeting the TBT SLO by allocating just enough SMs to the Decode phase.**
- **The remaining SMs are assigned to Prefill to maximize its execution speed, thereby achieving high throughput while maintaining SLO guarantees.**

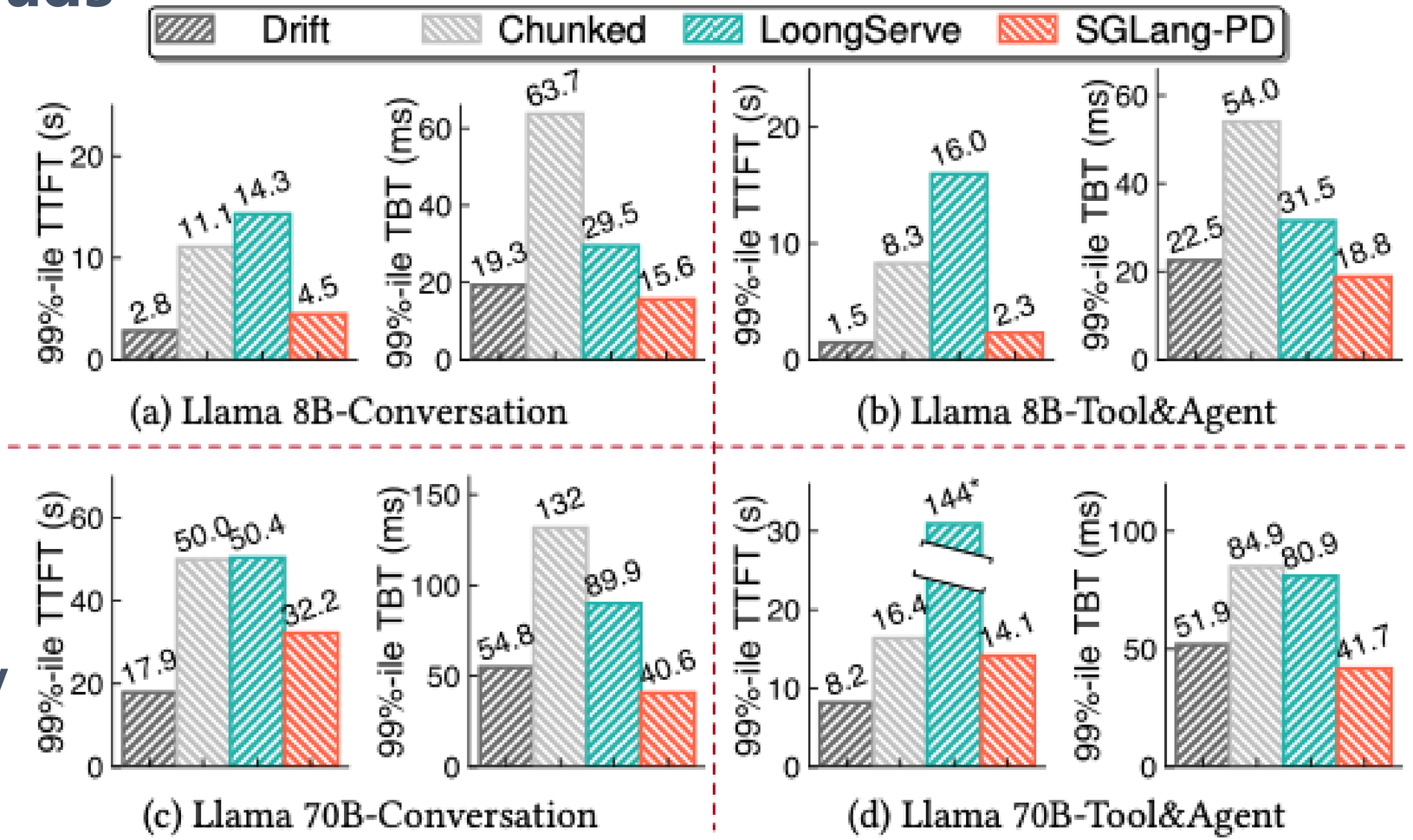
Evaluation

✦Real-world workloads

- ✦ Setting:
  8×A100-SXM4-80GB
- ✦ Trace:
  Mooncake Trace

+3.29×

While guaranteeing TBT SLO, significantly accelerate P99 TTFT.



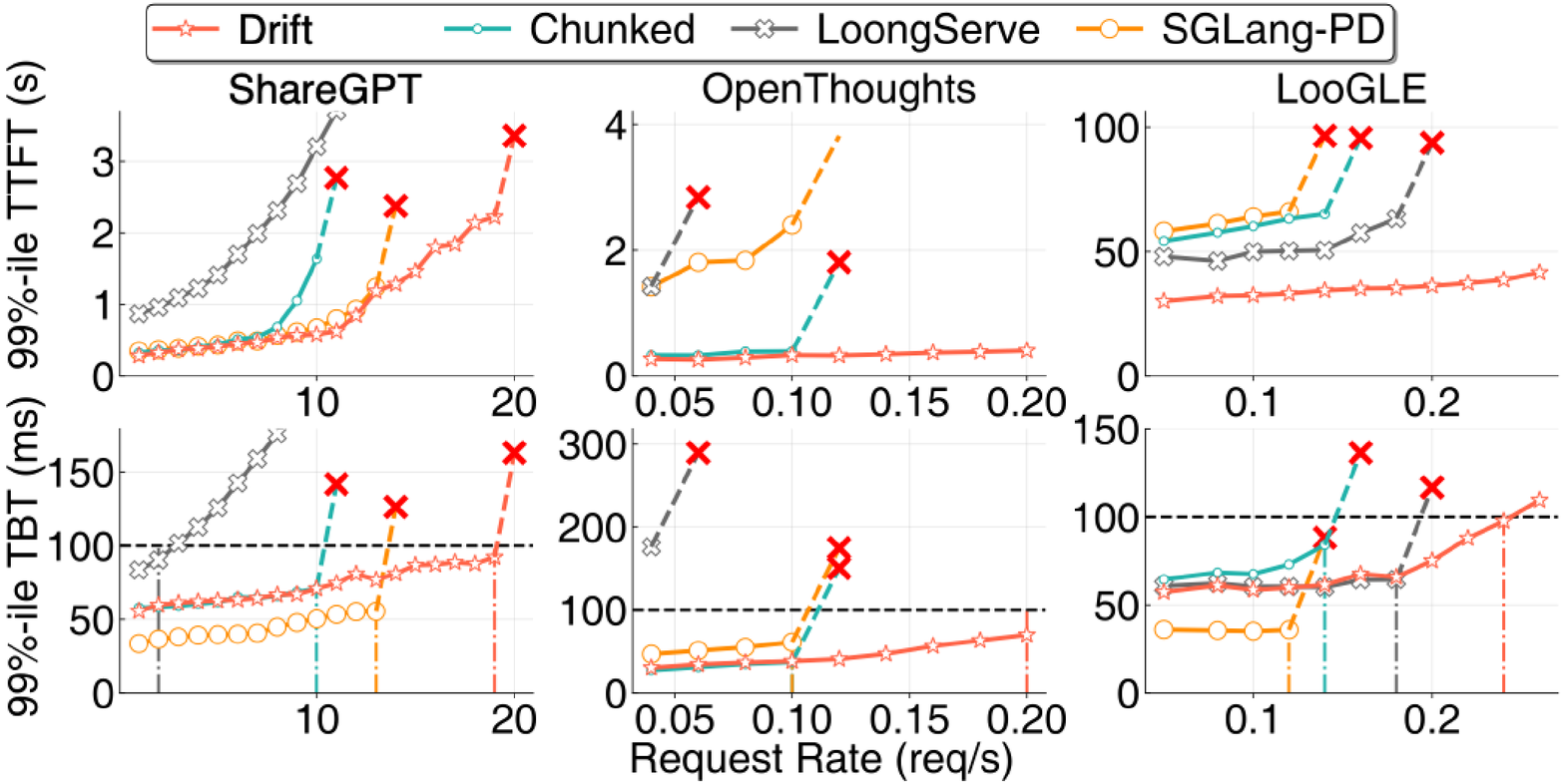
Evaluation

✦Benchmarks with different input/output lengths

- ✦ ShareGPT：similar in and out
- ✦ OpenThoughts：short in, long out
- ✦ LooGLE：long in, short out

+2.74×

With a 100 ms TBT SLO target, the average effective throughput is significantly improved



Thank you!