MLSys 2025

The Eighth Annual Conference on Machine Learning and Systems

SOLA: Optimizing SLO Attainment for Large Language Model Serving with State-Aware Scheduling

Ke Hong[12], Xiuhong Li[2*], Lufang Chen[2], Qiuli Mao[2], Xuefei Ning[1],

Guohao Dai[23*], Shengen Yan[12], Yun Liang[4], Yu Wang[1*]

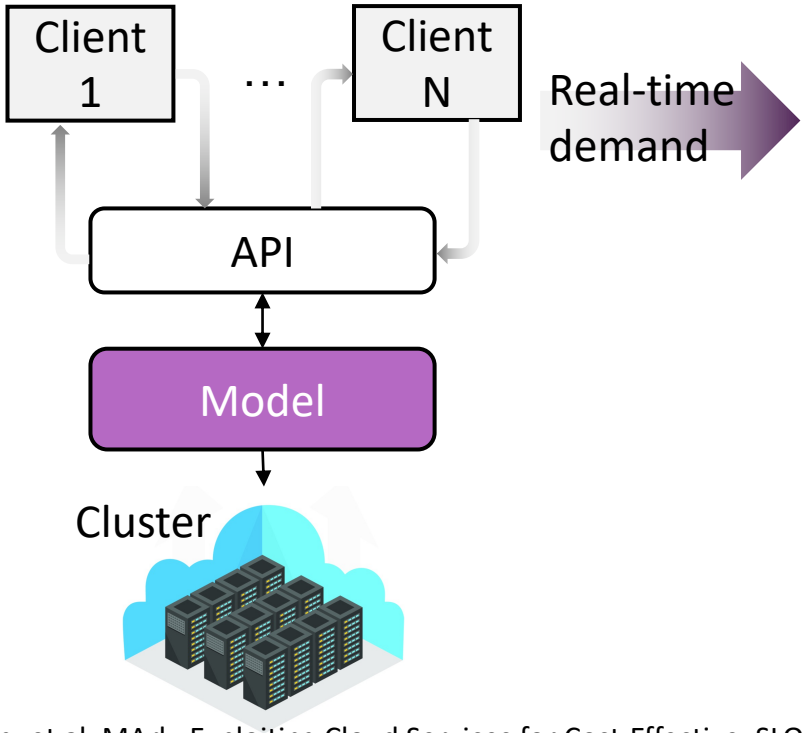
[1]Tsinghua University, [2]Infinigence-AI

[3]Shanghai Jiao Tong University, [4]Peking University

*Corresponding Authors

Background

Model as a Service (MaaS) and Service-Level Objective (SLO)



Real-time demand

An example of SLO: (**latency**, **attainment**)
99% *of requests must be served in* ***600ms***

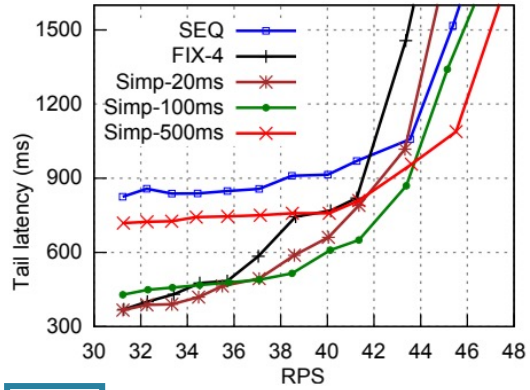


Figure 4: 99th percentile tail latency of sequential, degree 4, and simple fixed addition of dynamic parallelism in Lucene.

[1] C. Zhang, et al. MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving, ATC'19.

[2] M. Haque, et al. Few-to-Many: Incremental Parallelism for Reducing Tail Latency in Interactive Services, ASPLOS'15.

Background

The Goal of Service Providers: Maximize RPS Adhering to SLO

An example of SLO: (**latency**, **attainment**)
99% *of requests must be served in* ***600ms***

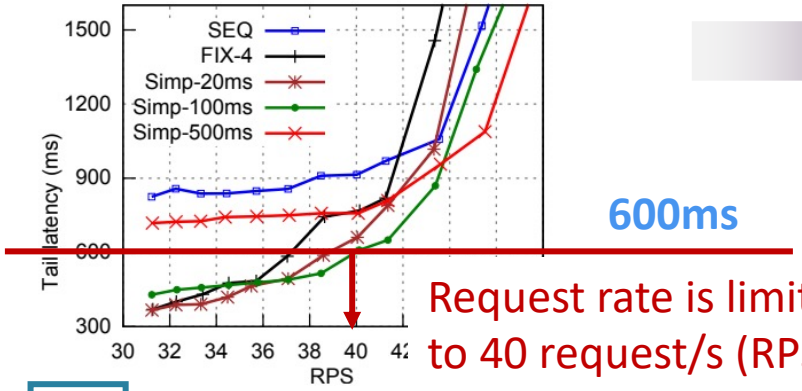
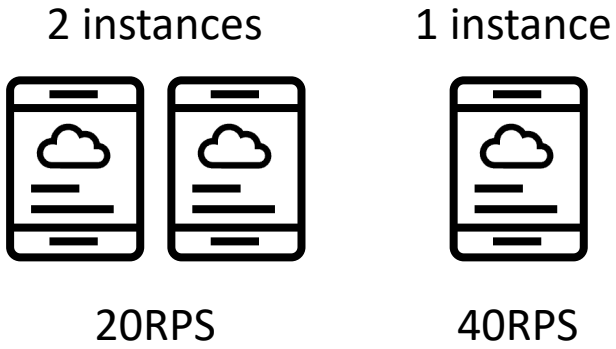


Figure 4: 99th percentile tail latency of sequential, degree 4, and simple fixed addition of dynamic parallelism in Lucene.

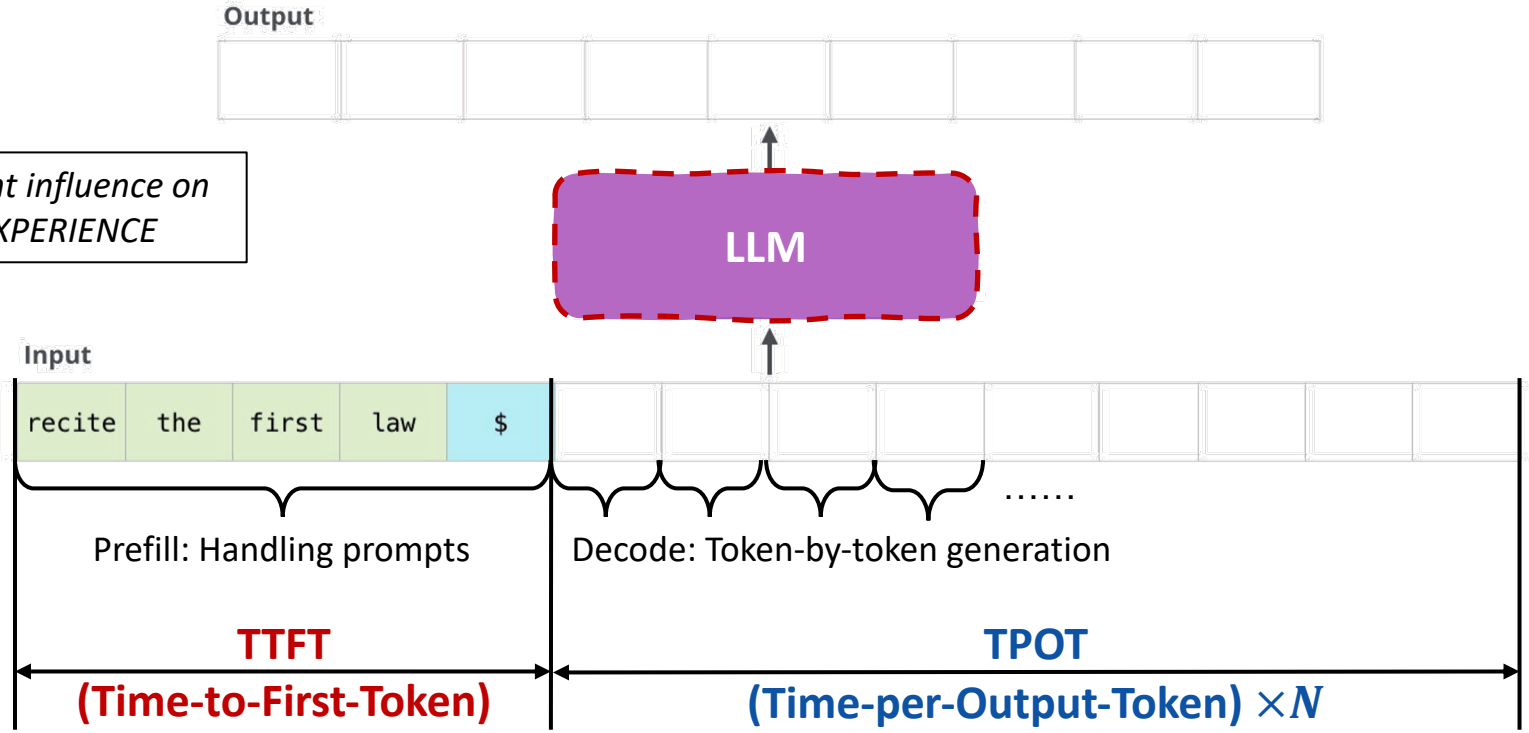
Higher RPS, Lower costs



Background

LLM Serving: Two Distinct Phases - Prefill and Decode

Different influence on USER EXPERIENCE

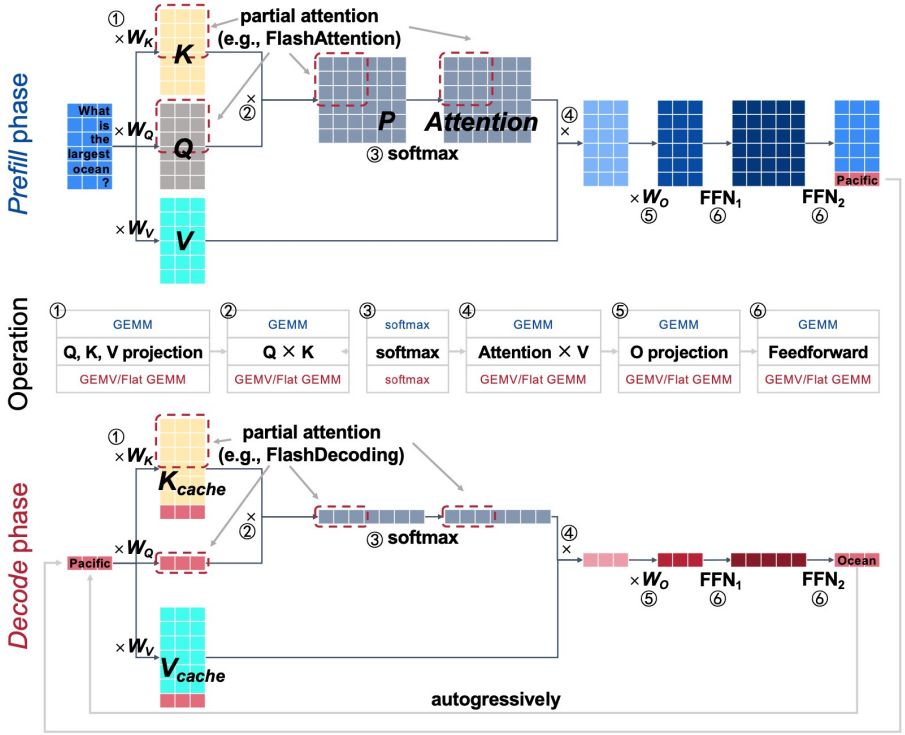


*Here LLM refers to the mainstream LLMs with Decoder-only Transformer architecture

Background

LLM Serving: Two Distinct Phases - Prefill and Decode

Totally Different COMPUTATIONAL CHARACTERISTICS



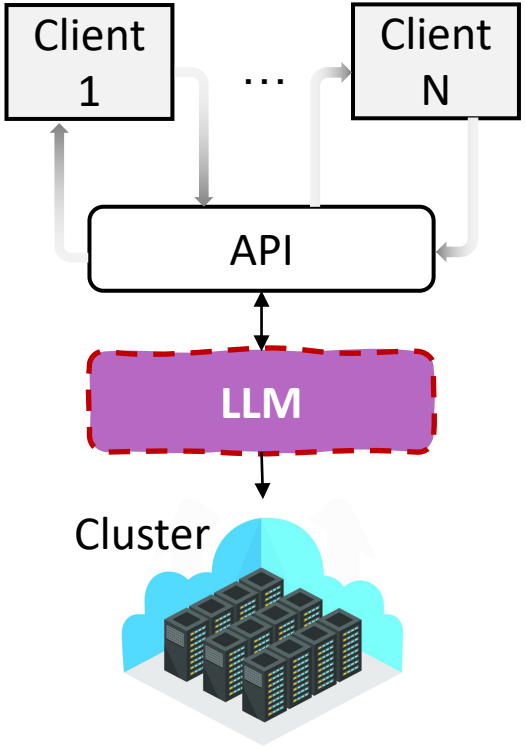
Prefill phase: Computation-intensive

Decode phase: Memory-intensive

[1]K. Hong, et al. FlashDecoding++: Faster Large Language Model Inference with Asynchronization, Flat GEMM Optimization, and Heuristics, MLSys'24.

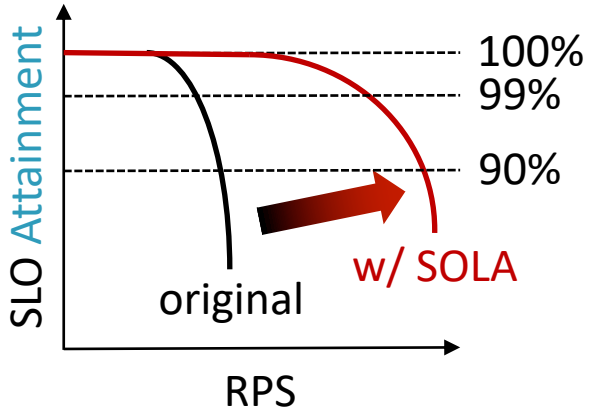
Overview

What We Do: Optimize SLO Attainment in Large Language Model (LLM) Serving



and

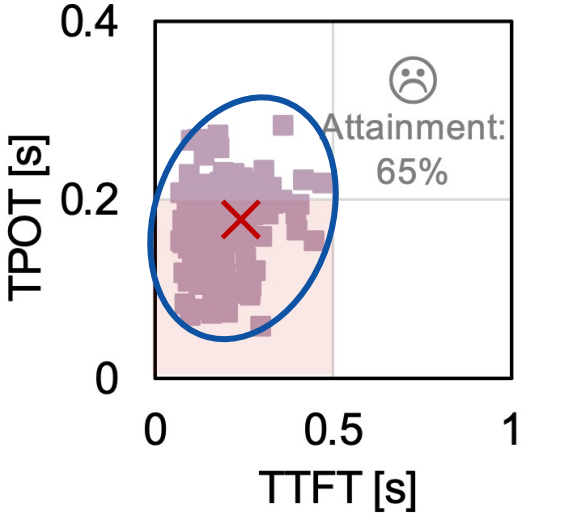
SLO: (**TTFT**, **TPOT**, **attainment**)



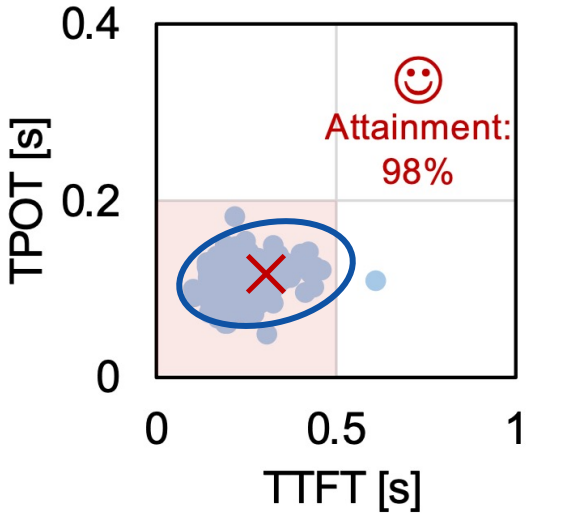
Given TTFT and TPOT,
higher attainment brings higher RPS

Motivation

The bias and the variance of the latency distribution



(a) Request distribution of vLLM



(b) Request distribution of SOLA

BIAS

one of TTFT and TPOT has much lower attainment against its SLO

VARIANCE

many requests fail the SLOs while others still having redundant budgets to spare

Motivation

Two-fold trade-offs exist in request-level latency

BIAS
one of TTFT and TPOT has much lower attainment against its SLO

→ ***Between TTFT and TPOT***

VARIANCE
many requests fail the SLOs while others still having redundant budgets to spare

→ ***Among different requests***

Motivation

Two-fold trade-offs exist in request-level latency

BIAS

one of TTFT
much lower
against its S

VARIANCE

many reque
while other
redundant budgets to spare

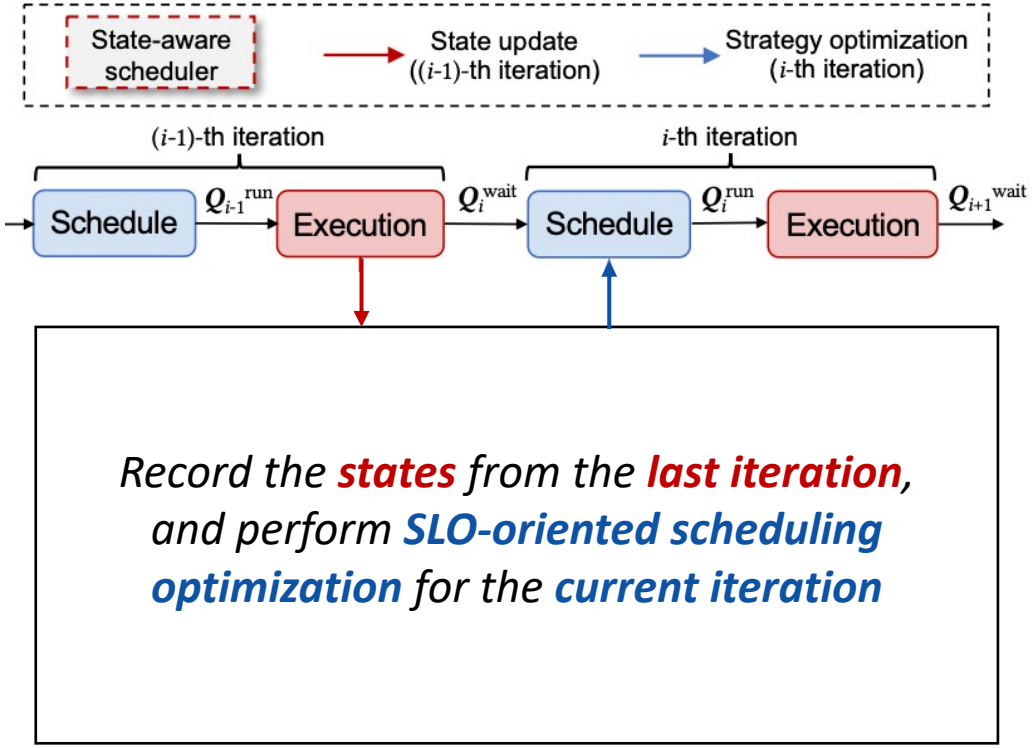
How to exploit such two-fold trade-offs?

nd TPOT

t requests

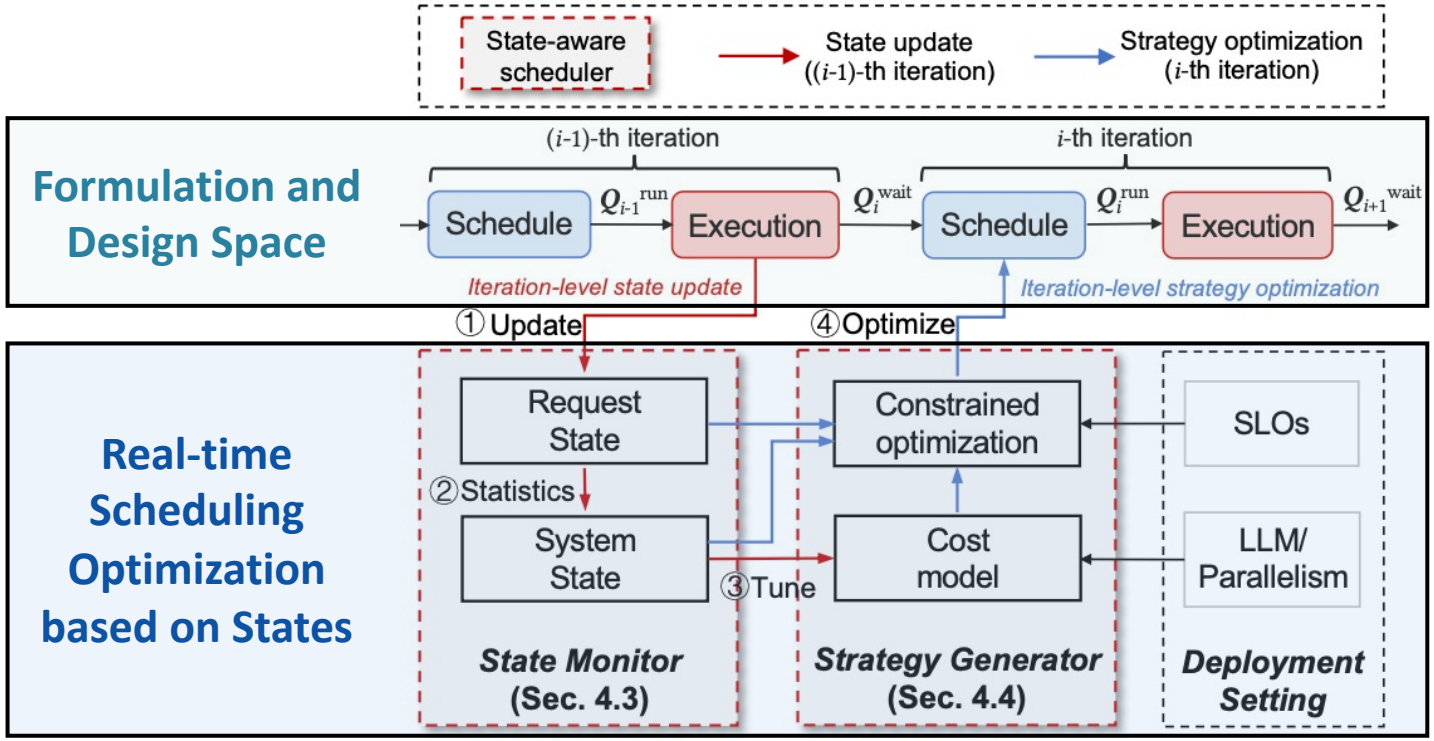
Overview

SOLA: State-Aware Scheduling



Overview

SOLA: State-Aware Scheduling

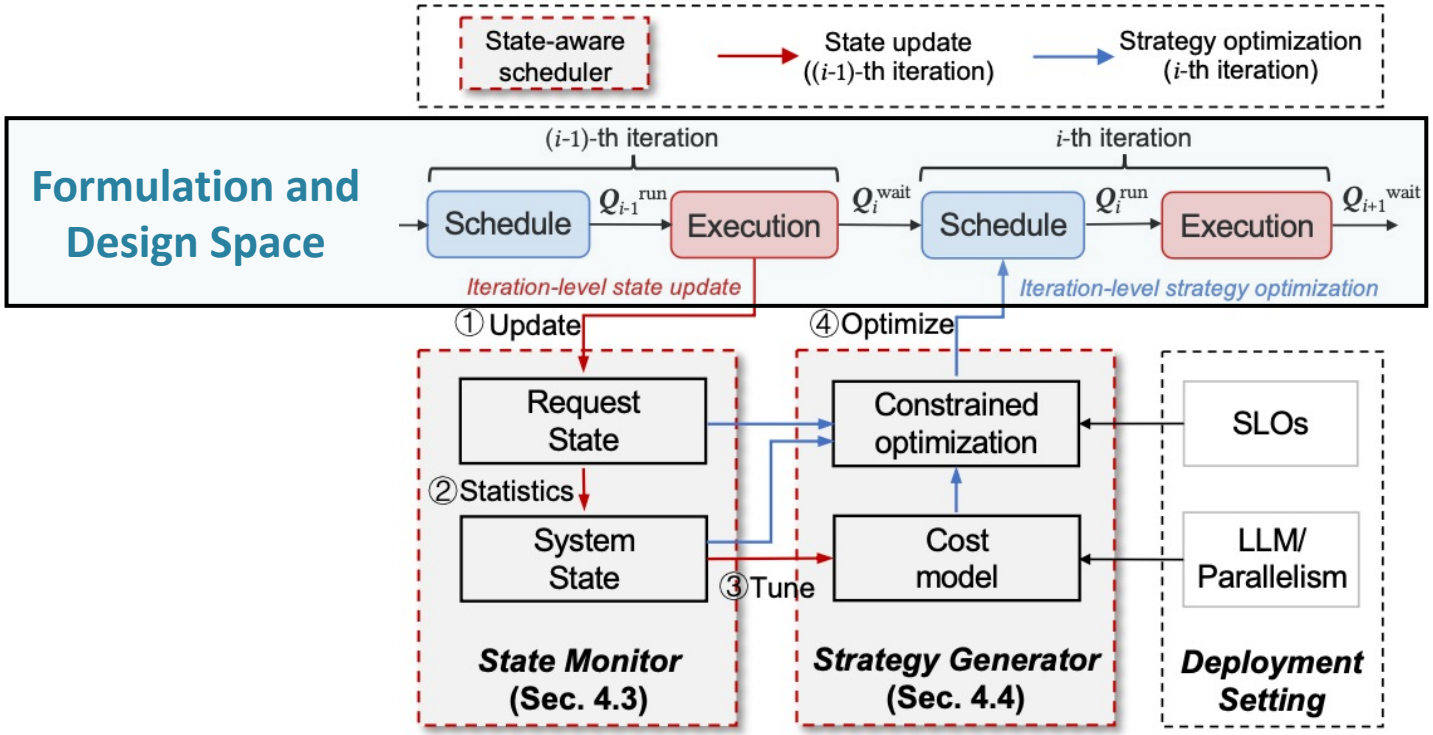


Iterative-level scheduling optimization

Aware of both system and request states

Overview

SOLA: State-Aware Scheduling



Iterative-level scheduling optimization

Modeling and Design Space

Algorithm 1 Scheduling at the i-th iteration.

Input: M, $\boldsymbol{Q}_i^{\text{wait}}$, $\boldsymbol{Q}_i^{\text{run}}$, m_i^{ratio}, n_i, k_i, $\mathcal{F}_i$

Initialize $\boldsymbol{Q}_i^{\text{run}} \leftarrow \emptyset$
if get_req_num($\boldsymbol{Q}_i^{\text{wait}}$) $= 0$ **then**
  Return ($\boldsymbol{Q}_i^{\text{wait}}$, $\boldsymbol{Q}_i^{\text{run}}$)
end if
$\boldsymbol{Q}_i^{\text{wait}} \leftarrow \mathcal{F}_i(\boldsymbol{Q}_i^{\text{wait}})$
for $r \in \boldsymbol{Q}_i^{\text{wait}}$ **do**
  $m^{\text{peak}} \leftarrow$ cal_peak_mem($\boldsymbol{Q}_i^{\text{run}} \cup r$)
  if $m^{\text{peak}} \geq M \times (1 - m_i^{\text{ratio}})$ **then**
    continue
  end if
  $\boldsymbol{Q}_i^{\text{run}} \leftarrow \boldsymbol{Q}_i^{\text{run}}$.push_req_with_token_num(r, $k_{i,r}^{\text{new}}$)
  $\boldsymbol{Q}_i^{\text{wait}} \leftarrow \boldsymbol{Q}_i^{\text{wait}}$.pop_req_with_token_num($r$, $k_{i,r}^{\text{new}}$)
  if get_req_num($\boldsymbol{Q}_i^{\text{run}}$) $\geq n_i$ **then**
    break
  end if
  if get_token_num($\boldsymbol{Q}_i^{\text{run}}$) $\geq k_i$ **then**
    break
  end if
end for
Return ($\boldsymbol{Q}_i^{\text{wait}}$, $\boldsymbol{Q}_i^{\text{run}}$)

Design space of scheduling

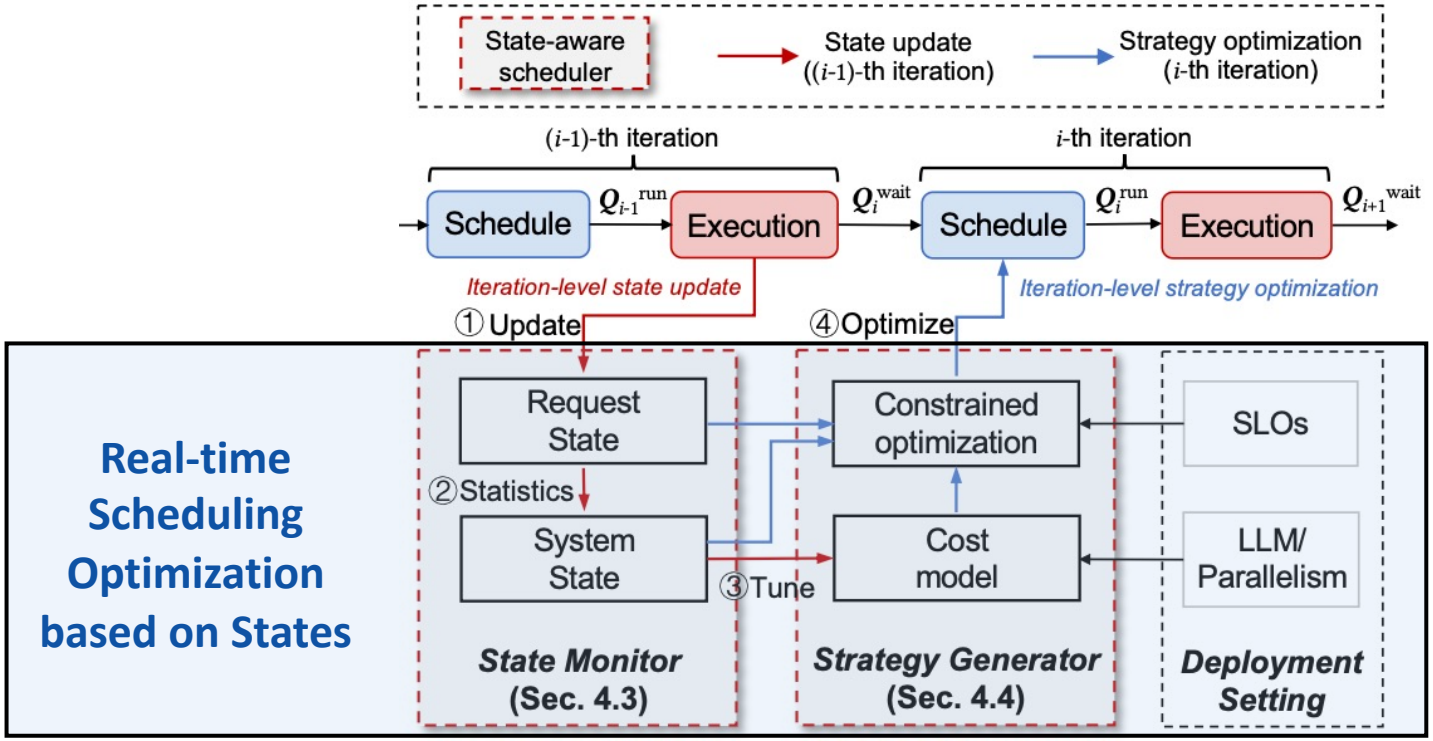
Execution Order
Which requests are prioritized

×

Workload Size
How many requests (or tokens) to run in this iteration

Overview

SOLA: State-Aware Scheduling



Real-time Scheduling Optimization based on States

Aware of both system and request states

Constrained Optimization

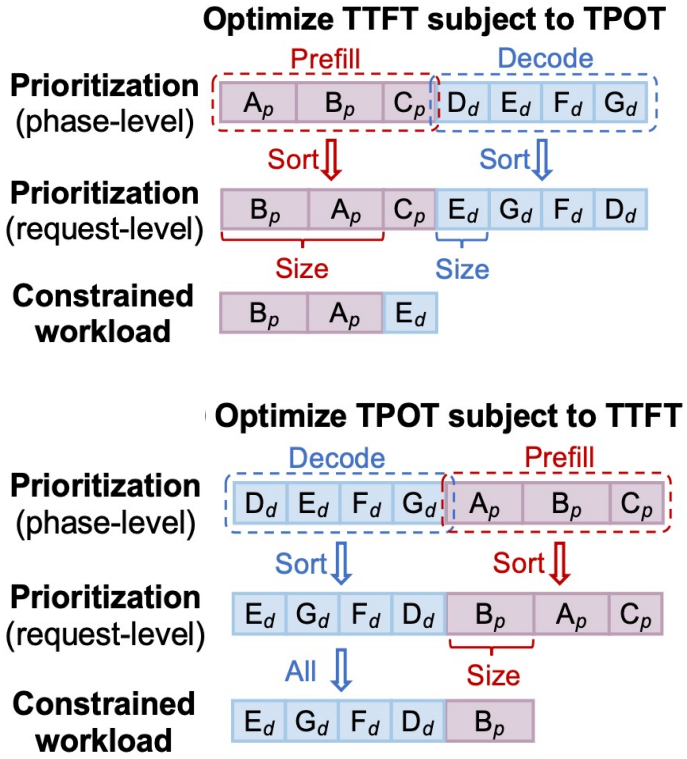
Problem: Constrained Optimization

Condition (or)	Constrained Optimization
(1) $1 > p_i^{\text{TPOT}} > p_i^{\text{TTFT}}$ (2) $p_i^{\text{TPOT}} > 1 > p_i^{\text{TTFT}}$	$\min \max_r(t_{i,r}^{\text{TPOT}})$ $s.t.\ \max_r(t_{i,r}^{\text{TTFT}}) \leq T^{\text{TTFT}}$
(1) $1 > p_i^{\text{TTFT}} > p_i^{\text{TPOT}}$ (2) $p_i^{\text{TTFT}} > 1 > p_i^{\text{TPOT}}$	$\min \max_r(t_{i,r}^{\text{TTFT}})$ $s.t.\ \max_r(t_{i,r}^{\text{TPOT}}) \leq T^{\text{TPOT}}$

$p_i^{\text{TTFT/TPOT}}$: the ratio between maximum real-time TTFT/TPOT and its SLO

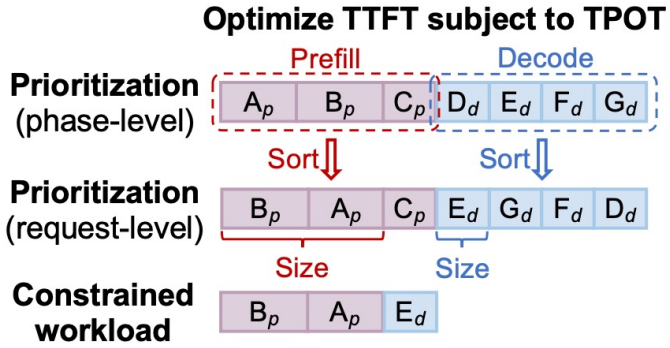
- **Objective:** the less fulfilled one of TTFT and TPOT SLOs, "**doing the best**"
- **Constraint:** the other one of TTFT and TPOT SLOs, "**ensure the constraint is met**"

Solution: Order and Workload Control

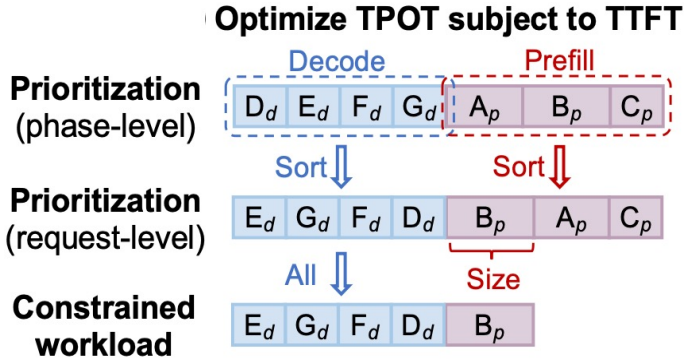


Constrained Optimization

	Order Control	Workload Control
Optimize TTFT subject to TPOT	*Prioritize prefill requests*	*Add as many as prefill requests until the predicted TPOT exceeds SLO*
Optimize TPOT subject to TTFT	*Prioritize decode requests*	*Retain as many as prefill requests in waiting until the predicted TTFT exceeds SLO*



Based on the predicted TTFT

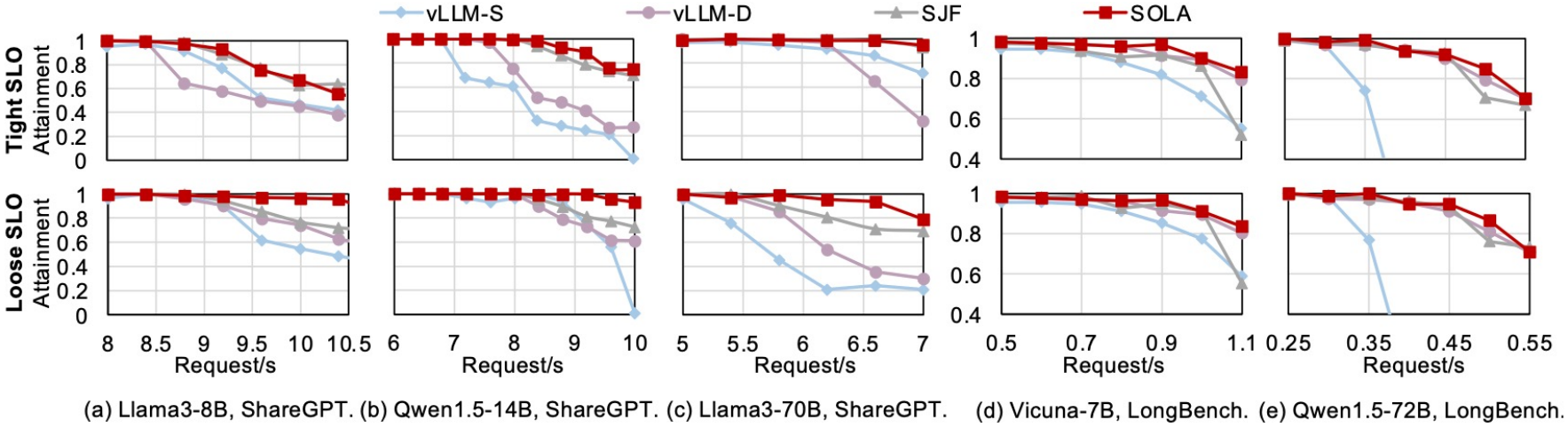


Based on the predicted TPOT

End-to-End SLO Attainment

Results1: Higher SLO Attainment

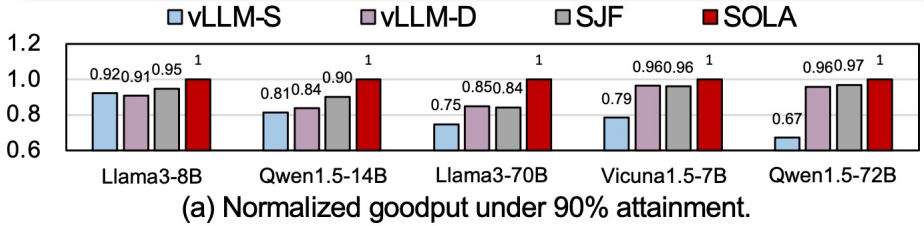
- **Baseline:** vLLM-S (vLLM with SplitFuse strategy), vLLM-D (vLLM with default strategy), SJF (Shortest-Job-First strategy)
- **Testbed:** NVIDIA A100 GPUs
- **Benchmark**: ShareGPT and LongBench datasets



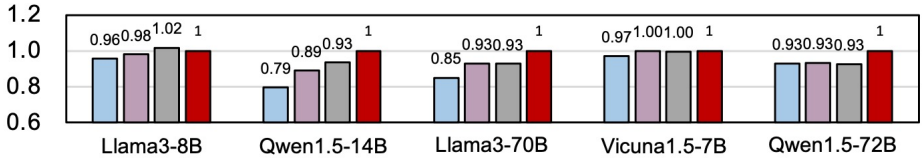
(a) Llama3-8B, ShareGPT. (b) Qwen1.5-14B, ShareGPT. (c) Llama3-70B, ShareGPT. (d) Vicuna-7B, LongBench. (e) Qwen1.5-72B, LongBench.

Other Results

Results2: 1.08-1.27x higher RPS adhering to SLO



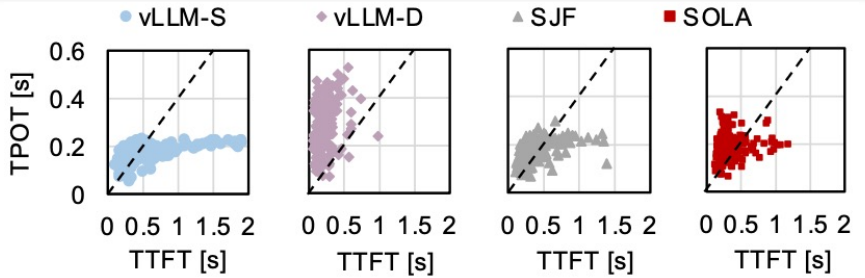
(a) Normalized goodput under 90% attainment.



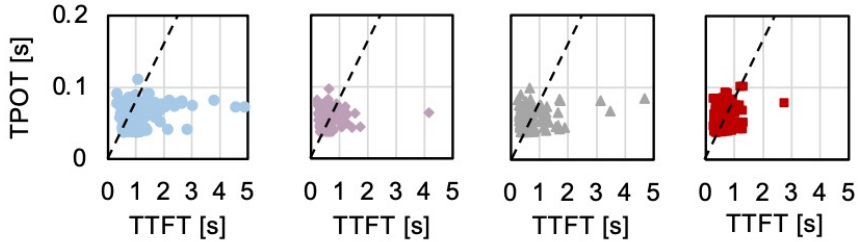
(b) Normalized goodput under 99% attainment.

- **SLO setting:** 10×/15× of the single request latency to be the tight/loose SLOs for 7B-14B models, and 5×/10× of the single request latency to be the tight/loose SLOs for 70B+ models

Results3: TTFT-TPOT plane visualization



(a) Llama3-70B, ShareGPT, 5.8 request/s.



(b) Qwen1.5-72B, LongBench, 0.35 request/s.

- More compact distribution with the center close to linearly scaled SLO line

Takeaway

The goal of this work:

Higher SLO attainment, higher RPS, lower costs

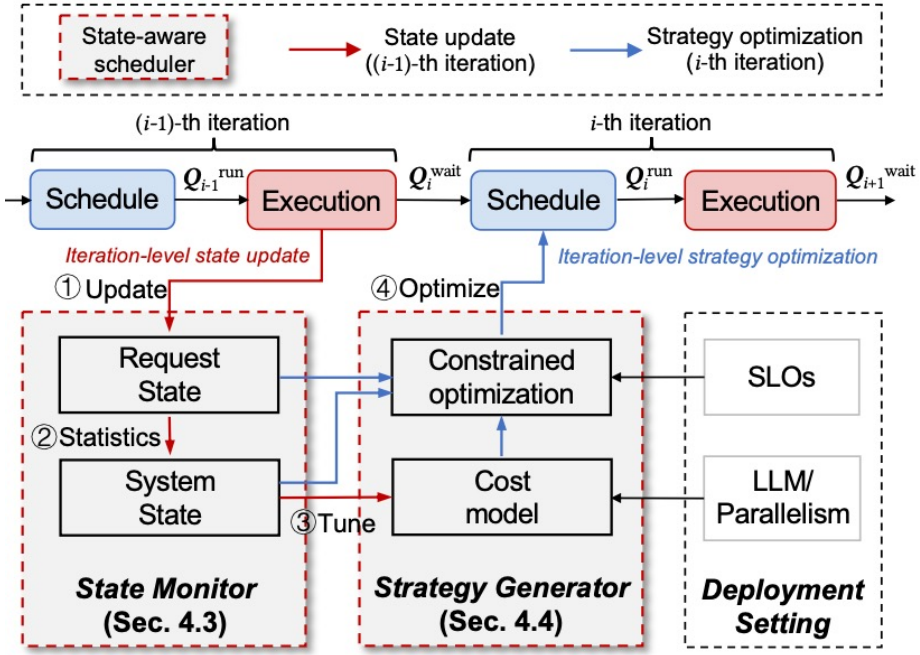
The motivation of this work:

We can exploit two-fold trade-offs (between TTFT and TPOT, among different requests) to optimize SLO attainment in LLM serving

What we do in this work:

1. ***The formulation of iterative-level scheduling and design space***
2. ***State-aware scheduling that is aware of both system and request states***
3. ***Constrained optimization on the scheduling strategy***

SOLA: Optimizing SLO Attainment for Large Language Model Serving with State-Aware Scheduling



Thank you, Q&A

Ke Hong, hk24@mails.tsinghua.edu.cn